Bayesian Economic Analysis for Software Release Planning under Uncertainty

Olawole Oni, Emmanuel Letier

Abstract—Release planning —deciding what features to implement in upcoming releases of a software system— is a critical activity in iterative software development. Many release planning methods exist but most ignore the inevitable uncertainty of estimating development effort and future business value. The few release planning methods that analyse uncertainty assume fixed-scope releases which is inconsistent with common industrial practices. This paper introduces BEARS, a novel release planning method under uncertainty that supports the common industrial practice of release cycles with fixed dates and flexible scopes. Experiments on 32 release planning problems drawn from 8 product backlogs show that BEARS recommends better release plans than methods that ignore uncertainty and than previous methods that deal with uncertainty. The paper provides empirical evidence of the benefits of analysing uncertainty during software release planning.

Index Terms—Release Planning, Requirements Prioritization, Bayesian Decision Analysis in Software Engineering, Value-Based Software Engineering, Search-Based Software Engineering, Software Economics

1 Introduction

Following an iterative software development process is widely recommended when requirements are uncertain and the project is subject to significant risks [1], [2]. Instead of delivering all software features at once, the software is delivered over multiple releases where each release adds or modifies features based on stakeholders' feedback and lessons learnt from previous releases. By feature, here, we mean both functional features and quality improvements (e.g. better performance). Since not all features are delivered at once, an essential activity of iterative development consists in prioritizing what features to deliver in upcoming releases. This activity, known as *release planning*, is critical to deliver business value, manage stakeholders' expectations, control risks, and organise software development activities [3].

Many release planning methods exist (see [4], [5] for surveys). These methods involve evaluating candidate features against a set of criteria relevant to the project stakeholders. Commonly used criteria include development effort measured in man-days or story points, business value measured in abstract value points (e.g. a number from 0 to 9) or in monetary units (e.g. in $), and stakeholders' satisfaction measured as abstract scores (e.g. a number between 0 an 100) or using problem-specific metrics (e.g. the percentage of user queries responded to within a certain time). Different methods propose different sets of criteria, different ways to elicit scores from stakeholders, and different ways to combine the elicited scores into some overall metrics that ultimately inform the release planning decisions.

Empirical studies in industrial contexts have shown the many benefits of systematic release planning methods over unstructured, ad-hoc release planning [6], [7], [8]. By introducing structure to the decision process, release planning methods mitigate problems encountered with unstructured decisions such as the lack of clarity about decision objectives and candidate features, the late identification of feature dependencies, the difficulty of engaging key stakeholders in the decision process, and the uncontrolled interference of power, politics and self-serving interests. Furthermore, the studies shows that introducing structure also reduced the time and effort involved in making decisions.

So far, however, no research has compared different release planning methods. Would different release planning methods applied in the same context recommend the same release plans? If not, do some methods recommend better release plans than others? Our objective in this paper is to study whether analysing uncertainty during release planning leads to better decisions than if uncertainty is ignored.

Most release planning methods ignore the inevitable uncertainty of software development effort and value. They evaluate candidate release plans as if all features will be delivered on time according to plan. In reality, some features will take longer to develop than predicted and, once released, may deliver more or less business value than expected. Ignoring such uncertainty can lead to inaccurate, overoptimistic, and inconsistent evaluation of release plans, which in turn can lead to misinformed and possibly inadequate release planning decisions.

A few methods have been proposed to analyse uncertainty during release planning (e.g. [9], [10], [11], [12], [13]) but they suffer two important limitations:

1) No evidence exists that the added complexity of analysing uncertainty leads to better decisions than simpler methods that ignore uncertainty. Without such evidence, the added complexity cannot be justified and release planning methods under uncertainty are unlikely to be adopted.
2) They rely on assumptions that are inconsistent with the most common current industrial practices. Surveys of industrial practices indicate that most organisations follow a release process where the release dates are fixed (e.g. every 3 months) but the content

of each release is flexible, i.e. the release content may differ from what was planned [8]. Existing release planning methods under uncertainty, however, assume the opposite: release contents are fixed and only the release dates are flexible. The difference has implications on what uncertainty needs to be reasoned about: uncertainty about release dates for fixed-scope releases, uncertainty about release contents for fixed-date releases. Release plans that are optimal assuming fixed-scope releases may not be optimal under a release process with fixed-date, flexible-scope releases.

This paper address these limitations by:

1) introducing BEARS, a novel release planning methods under uncertainty designed for the common industrial practice of fixed-date, flexible-scope releases;
2) presenting experiments on 32 release planning problems showing that analysing uncertainty using BEARS leads to shortlisting better release plans than if uncertainty is ignored or if uncertainty is analysed assuming fixed-scope releases.

BEARS is an acronym for "Bayesian Economic Analysis of Release Scenarios". The acronym is also a nod to the seminal book, "Waltzing with Bears: Managing Risks on Software Projects", one of the first books to argue for embracing uncertainty in software projects [14]. BEARS is a Bayesian method in the sense that it uses Bayesian probability to model the release planners' subjective uncertainty about the development effort and business value of candidate features. BEARS is supported by a tool that takes as input subjective probabilities of the effort and value of individual features and generates as output a shortlist of Pareto-optimal release plans that maximize expected net present value (a common financial metric for comparing flows of values) and maximize expected punctuality (defined as the expected percentage of features delivered on time). These two objectives are conflicting: maximizing expected net present value without consideration for punctuality leads to selecting over-ambitious release plans that are unlikely to be completed in time, whereas maximizing expected punctuality without consideration for value leads to selecting over-conservative release plans with low expected value. BEARS's output is a set of release plans that shows all possible optimal traedoffs between these two extremes. Release planners can then inspect the shortlisted plans to select one that best suits their preference. The BEARS tool and all data needed to replicate our experiments are available from the tool's public repository [1].

2 Motivating Example

Throughout the paper, we illustrate and compare alternative release planning methods using an example based on a real local government project. The project aims to develop online services allowing local residents and businesses to perform various tasks such as paying local taxes, reporting missed rubbish collection, managing parking permits, and submitting and responding to planning applications online instead of over the phone or by visiting offices in person. Many residents and businesses have expressed preferences for performing such transactions online when possible. Each online interaction is much cheaper to the local government than the equivalent interaction over the phone or in person. The local government has identified over 20 individual services that could be delivered online but its Information System group has limited resources and is only able to develop a few features supporting these services every quarter. Different stakeholders disagree about which features should be developed first. The project manager and software development team need to prepare a release plan that will organise the development work so that it provides the most value to the local government and its residents.

1. https://github.com/olawole/BEARS

3 Release Planning: Background

This section provides a brief introduction to key concepts and methods of software release planning.

3.1 Product Backlog and Release Plans

Release planning takes as input a backlog of work items to be scheduled for future releases and generates as output a release plan specifying what items are scheduled for what future releases.

Definition (Product Backlog): A *product backlog* is

- a set *WI* of work items that are candidates for future releases; and
- a relation $\leftarrow \subseteq WI \times WI$ that defines a precedence relation between work items, i.e. $w_i \leftarrow w_j$ means that work item w_i must be delivered before or at the same time as work item w_j.

In some release planning methods, backlog items are called requirements [15], [16], in others they are called features [3], [17], [18]. In this paper, we will follow the terminology of the Incremental Funding Method where work items are either features or architectural elements [17]. Features are functional or quality improvement that deliver value to stakeholders; *architectural elements* are software elements that do not in themselves deliver value to stakeholders but are prerequisites to the development of features.

In addition to precedence relation, the product backlog may include other types of dependencies between work items such as coupling, exclusion, and weak precedence [3], [19]. We will not use these additional dependencies in this paper; all techniques described in the paper can however be extended to handle these additional dependency types (the required changes will be discussed in Section 4.6).

As an example, Figure 1 shows a small fragment of the product backlog for our local government project. The full backlog, available in BEARS's online repository, is composed of 15 features, 5 architectural elements, and 23 precedence links.

The main output of release planning is a release plan.

Definition (Release Plan): Given a product backlog $< WI, \leftarrow >$, a release plan is a partial function $p : WI \rightarrow [1..H]$ that maps work items to future releases. The number $H \in \mathbb{N}^+$ of releases in the plan is called the *planning horizon*.

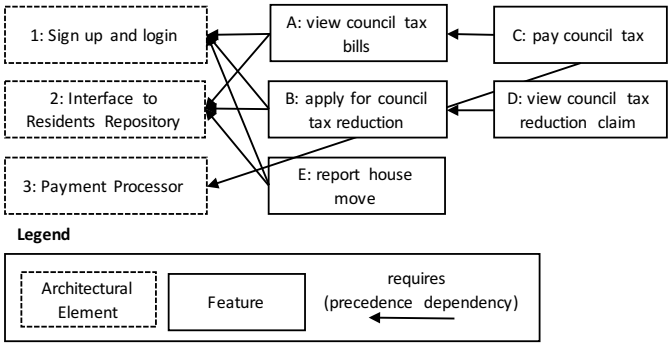


Fig. 1: Part of the product backlog for the local government project

TABLE 1: Example effort estimation in man-days

Work Items	Effort
A: view council tax bills	2
B: apply for council tax reduction	3
C: pay council tax	2

For a work item w, $p(w) = i$ means that w is planned to be delivered in the i^{th} next release. A release plan is a partial function because not all work items in the backlog need to be planned for some future release. Release plans must satisfy the precedence constraints: if $w_i \leftarrow w_j$ then $p(w_i) \leq p(w_j)$. Each release also has a planned release date, *releaseDate* : $[1..H] \rightarrow$ *Date*. Many organisations use a fixed release cycle where successive release dates are separated by a constant time interval (e.g. 3 months) [4], [5], [8].

3.2 Managing Capacity Constraints

Release planning must ensure that the effort required to develop the work items planned for each release does not exceed the development team's capacity. To model this constraint, the product backlog is extended with the following information:

- *effort*$(w) \in \mathbb{N}^+$ denoting the estimated effort required to deliver each $w \in WI$;
- *capacity*$(i) \in \mathbb{N}^+$ denoting the estimated team capacity in each release period $i \in [1..H]$.

Effort and capacity estimates are provided by the development team and release planners. Techniques to support such estimations include planning poker [20] and formal estimation models [21]. Effort and capacity must be expressed in the same unit, for example in man-days or story points. Story points are popular in lean and agile software development. They denote abstract scores on a ratio scale (e.g. a score between 0 and 100) that denote the *relative* effort needed to deliver a work item in comparison to other work items (e.g. an item worth 50 story points is expected to require 5 times the effort of an item worth 10 story points). Effort estimation in man-days denote, as the name implies, the estimated number of man-days required to deliver an item.

As an example, Table 1 shows effort estimates in man-days for 3 features in our local government project.

Release plans must satisfy the constraint that the effort to develop all work items in a release period does not exceed the team capacity for that period, i.e. for all $i \in [1..H]$:

$$\sum_{\{w \in WI \mid p(w)=i\}} \textit{effort}(w) \leq \textit{capacity}(i) \quad (1)$$

Constraints on other resources such as personnel, equipments, and capital can be modelled in the same way by specifying the resource requirements for each work item and the resource capacity during each period [3].

A well-known problem in managing capacity constraints is the difficulty of predicting development effort: effort estimates are often optimistic and ignore the inevitable uncertainty of software development tasks [22]. Ignoring such uncertainty, or at least not describing it explicitly, prevents release planners from analysing the likelihood and impacts of late deliveries.

Some release planning methods address this problem by estimating effort as probability distributions [9], [10], [11], [23], [24]. Effort uncertainty can for example be modelled by triangular distributions using the most optimistic, most likely and most pessimistic effort estimates for each work item [9], [11], [23], [24]. Other methods model such uncertainty using lognormal distributions [10]. Given these probability distributions, the above methods aim to maximise the probability that the next release is delivered within budget [23], [24] or by its planned date [10], [11]. These methods, however, are limited to analysing a single next release (i.e. their planning horizon H is limited to 1). Another method, dealing with multiple releases, aims to maximize the probability that all planned releases are delivered on time [9]. This probability is formally defined as:

$$P(\forall i \in [1..H] : \sum_{\{w \in WI \mid p(w)=i\}} \textit{effort}(w) \leq \textit{capacity}(i)) \quad (2)$$

We will use this method as point of comparison in our evaluation in Section 5.

Note that all the above methods assume *fixed-scope* releases, i.e. a new release is delivered only when *all* features planned for that release have been completed. They only allow for flexibility in the release dates and budgets, not in their scopes. As mentioned earlier, surveys have shown that fixed-scope releases are rarely used in practice; many organisations preferring to use a fixed-date release cycle where new releases are delivered at regular intervals (e.g. every 3 months) and include all features that are ready by the planned release date [8].

3.3 Optimizing Value Points

Software systems are built to deliver value to their stakeholders [25]. To help release planners identify which release plans are likely to deliver most value, many release planning methods rely on assigning value points to release plans. In these methods, value points are abstract scores denoting *relative* value on a ratio scale (e.g. between 0 and 100); they do not correspond to concrete observable quantities (e.g. financial gain, number of users of the online services).

The product backlog is extended such that each work item w has a value point score, noted *valuePoints*(w), denoting the relative value of w in comparison to other work items. The 'Business Value' attribute used in project management systems such as JIRA and Visual Studio Team Services are typical example of such value points. Value points

are to be provided by release planners, product owners or project stakeholders.

In a release plan p, the values points of release i is the sum of the value points of all work items planned for that release:

$$valuePoints(p,i) = \sum_{\{w \in WI \mid p(w)=i\}} valuePoints(w) \quad (3)$$

The value points of release plan p is the weighted sum of the value points in each release:

$$valuePoints(p) = \sum_{i=1}^{H} weight(i) \times valuePoints(p,i) \quad (4)$$

where $weight(i)$ denotes the weight given to value points in release i. The release weights, to be specified by the release planners, denote the relative importance of value points delivered in earlier releases over those delivered in later releases. Formally, the weights define marginal rates of substitution between value points in different releases. For example, specifying weights 5, 3, 1 for the first three releases mean that 1 value point in the first release is equivalent to 5/3 value points in the second release and 5 points in the third release.

The release planning method EVOLVE-II extends this model by deriving work items' value points from the assessment of each work item by multiple stakeholders according to multiple criteria [3]:

$$valuePoints(w) = \sum_{c \in C} weight(c) \times (\sum_{x \in S} weight(x) \times score(w,x,c)) \quad (5)$$

where

- C is a set of criteria and *weight(c)* is the weight of criteria c;
- S is a set of stakeholders and *weight(x)* the weight of stakeholder x;
- *score(w, x, c)* denotes the score assigned by stakeholder x to work item w for criteria c.

For example, Table 2 illustrates how EVOLVE-II would elicit value points for three features in our motivating example. The two criteria of interests are 'Frequency' denoting how often a service is used and 'Savings' denoting how much savings would be made by delivering this feature. The two stakeholders' groups are the local council residents and staff. Each stakeholder group is asked to score each work item against each criteria on a scale from 0 to 9. The work items' value points are then computed according to Equation 5.

EVOLVE-II then uses an optimisation algorithm to shortlist the top 10 release plans that maximize value points (Eq. 4) while satisfying the capacity constraints (Eq. 1). Release planners can then inspect that shortlist to select their preferred release plan.

The EVOLVE method has many variants (at least 22 according to two recent surveys [4], [5]). Most variants

TABLE 2: Eliciting value points in EVOLVE II

Work Item	Frequency		Savings		ValuePoints
	Residents	Staff	Residents	Staff	
A: view council tax bills	5	4	2	6	4
B: apply for council tax reduction	7	5	3	8	5
C: pay council tax	5	5	6	9	7

Stakeholders Weights: Residents = 0.6, Staff = 0.4
Criteria Weights: Frequency = 0.3, Savings = 0.7

amounts to pre-selecting the criteria to be used in Equation 5. Examples of such criteria include urgency, risk, stakeholder satisfaction, competitiveness, and cost of delay. All such criteria are still evaluated as abstract scores and aggregated through weighted sums, although some variants use more complex formulae.

One interesting extension consists in analysing the values of release plans for different stakeholders' groups separately [26]. This allows release planners to explore tradeoffs between satisfying different stakeholders' groups and to analyse the fairness of candidate release plans with respect to the different groups.

An important, often overlooked limitation of release planning methods using value points is the difficulty of eliciting scores and weights that accurately reflect the stakeholders' true preferences. In Equations 4 and 5, the weights and scores correspond to marginal rates of substitutions. Eliciting such rates of substitutions is far from trivial and one can question whether the elicited numbers (such as those in Table 2) accurately reflect the stakeholders' and release planners true preferences. With inaccurate inputs, the evaluation and ranking of release plans may also be inaccurate.

3.4 Optimizing Net Present Value

An alternative to optimising abstract value points is to optimise release plans according to financial metrics such as net present value and return on investment [17], [27], [28].

The Incremental Funding Method (IFM) is the first method to use such economic approach [17]. It requires release planners to specify the projected cash flow of each work item, $cashFlow(w) : [1..L] \rightarrow \mathbb{R}$ where L is the investment horizon ($L \geq H$), i.e. the period over which the total value of release plans is measured, and $cashFlow(w,i)$ denotes the projected cost or revenue of w during the i^{th} period after the start of its development. For example, $cashFlow(w) = [-2000, 1000, 1000, 1000, 1000, 1000]$ means that developing w will cost \$2,000 during one period and once released will generate \$1,000 per period for the next five periods.

In the IFM, value is more easily defined over development plans than release plans. A development plan is a partial function $p : WI \rightarrow [1..H]$ that maps work items to the period in which their development starts. A development plan's cash flow is a function $cashFlow(p) : [1..L] \rightarrow \mathbb{R}$ computed from the work items' projected cash flow such that $cashFlow(p,i)$ is the sum of all work items' costs and revenues taking into account when their development started. The Net Present Value (NPV) of a development plan p is then defined as:

$$NPV(p) = \sum_{i=1}^{L} \frac{cashFlow(p,i)}{(1+r)^i} \quad (6)$$

where r is the discount rate. NPV is a standard financial metric to compare cash flows taking into account the time value of money at a given discount rate. The discount rate allows one to take into account that \$100 dollars today are worth more than \$100 dollars in a year. Note that the discount rate plays a similar role as the release weights in the previous section; it allows one to compare values delivered at different times.

Unlike EVOLVE, the IFM does not include effort constraints. Instead, it assumes the development team can only work on a bounded number of work items per period (e.g. at most 2 work items per period). The IFM uses heuristics to search for development plans that optimise NPV. The output of the IFM is a single development plan that has optimal or near optimal NPV. Release planners can use this plan as baseline to explore alternatives and select some preferred release plan.

Extensions to the original IFM include improved algorithms for identifying optimal development sequences [29], analysis of cash flows uncertainty [12], [13], and analysis of competitors' behaviours using game theory [30]. The IFM extensions dealing with cash flow uncertainty ignore uncertainty about development time and assume a fixed-scope release process. They also assume a single work item can be worked on at a time. Our new release planning method in Section 4 will address these limitations.

Arguments against such economic analysis are the inaccuracy of most cost and revenue projections, and their tendency to overlook important but not easily quantifiable values, related for example to stakeholders' satisfaction and other human, social and environmental values. An advantage over the value points approaches of the previous section, however, is that economic value can often be derived from observable factors. For example, in our local government project, the economic value of a feature can be evaluated by estimating how many phone interactions per months will be saved by introducing the feature and multiplying it by the average cost of a phone interaction. We argue such evaluation is more credible (or at least less arbitrary) than the weighted sum of scores between 1 and 9 in the previous section. Values that at first may seem hard to quantify, such as the residents satisfaction in our running example, can also often be related to observable factors (e.g. residents' ratings and complaints about the council's services) and measured in economic terms by eliciting stakeholders' *willingness-to-pay* for such values [31]. Such economic analysis are nevertheless fraught with uncertainty. The next section presents a release planning methods that takes such uncertainty into account.

4 Release Planning with BEARS

4.1 Overview

BEARS supports release planning under uncertainty in the context of release cycles with fixed date and flexible scope.

The main input to BEARS is a product backlog where each work item w has the following attributes:

- *effort(w)*: a real-valued probability distribution denoting the release planners' subjective uncertainty about the number of man-days required to deliver w;
- *value(w)*: a real-valued probability distribution denoting the release planners' subjective uncertainty about the value in monetary units brought about by w in each period after it is delivered.

BEARS also requires release planners to specify the following additional parameters:

- the planning horizon $H \in \mathbb{N}^+$ and the team capacity, noted *capacity(k)*, in each period $k \in [1..H]$;
- the investment horizon $L \in \mathbb{N}^+$ and the discount rate $r \in \mathbb{R}$ used to compute net present values;
- and, optionally, the budget B allocated to development team for the next H iterations. This is a fixed quantity to cover all costs for the fixed duration of the next H releases.

Given these inputs and parameters, BEARS evaluates candidate release plans against the following criteria:

- their *Expected Net Present Value (ENPV)*, and
- their *expected punctuality* (EP) defined as the expected percentage of work items delivered on time;

BEARS is supported by a tool that automatically shortlists a set of Pareto-optimal release plans that maximize expected NPV and expected punctuality. The tool is implemented in JAVA. It uses JMetal [32] for multi-objective optimisation and our own implementation of a Monte-Carlo simulation of release plans that will be described in Section 4.3. The tool takes as input a CSV file defining the work items' dependencies and effort and value estimates. It generates a CSV file with the shortlisted Pareto-optimal release plans and a figure of the Pareto front. It also computes and reports the expected value of perfect information, a form of probabilistic sensitivity analysis used to help release planners decide whether they should seek more information about some item's effort and value before making a decisions [33]. We do not elaborate on information value analysis in this paper.

4.2 Eliciting Effort and Value Uncertainty

Reliable, principled elicitation techniques exist for eliciting subjective uncertainty from persons and groups of persons [34]. These techniques assume that people have some real, tacit uncertainty about the quantities of interest (in our case work, the items' effort and value) that can be uncovered through targeted questions and modelled as Bayesian probability distributions. The elicitation techniques are designed to mitigate common estimation biases such as overconfidence and anchoring. We use the SHELF elicitation framework [34] and the associated MATCH tool [35] to perform such elicitation.

Currently, BEARS assumes effort and value have lognormal distributions, although we intend to support a wider range of distributions in the future. Lognormal distributions are appropriate to model uncertainty about quantities that are always positive and where the distribution can be asymmetric and have a long tail of possible high values.

As an example, Figure 2 illustrates the elicitation of uncertainty about the effort to develop feature A in the product backlog. The elicitation uses the 'quartile method' that consists in eliciting the development team's subjective

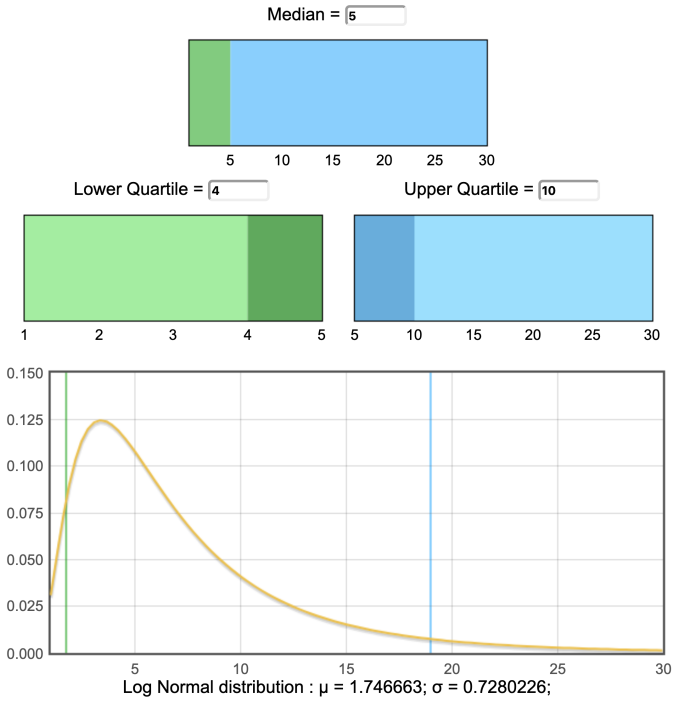


Fig. 2: Eliciting effort uncertainty for feature "A: view council tax bills" with the MATCH tool [35].

assessment of: (i) the effort's upper and lower limits denoting minimum and maximum plausible values; (ii) the effort's median value denoting the value M such that the development team judges it equally likely that the true effort is below M or above M; (iii) the effort's lower quartile denoting the value $Q1$ such that the development team judges it equally likely that the true effort is below $Q1$ or between $Q1$ and M; and (iv) the effort's upper quartile denoting the value $Q2$ such that the development team judges it equally likely that the true effort is above $Q2$ or between M and $Q2$. A facilitator helps the team members adjust their estimates by testing their indifference to alternative bets. For example, to test the median, the facilitator asks the development team to consider a bet where they would win a large sum of money if they can correctly guess whether the true effort will be either below or above the stated median. If they have a preference for betting either 'below' or 'above', then their stated median is not the true median of their subjective uncertainty and should be adjusted accordingly. Once all data points have been elicited, the MATCH tool infers the best-fit lognormal distribution as shown in Figure 2.

4.3 Simulating Release Scenarios

In order to simulate release plans, BEARS distinguishes *release plans* $p : WI \rightarrow [1..K]$ that specify when work items are planned for release from *release scenarios* $s : WI \rightarrow [1..K]$ that specify when work items are actually released. During release planning, the future actual release scenario is unknown.

To simulate release scenarios, BEARS first generates N simulations of work items' effort and value drawn from the effort and value probability distributions. By default, BEARS uses $N = 10^4$. This creates N possible future worlds, each having specific effort and value data for all work items. In each possible future world, BEARS creates the release scenario that will happen in this world based on the work item's effort data. This is done by first mapping the release plan to a work sequence that specifies in what order work items will be worked on (note that release plans do not specify any order between items to be delivered in the same release), then by generating the release scenario from the work sequence.

To generate a work sequence from a release plan, we apply the following priorities between work items: an item w_i takes precedence over an item w_j in the work sequence if: (i) w_i is planned for an earlier release than w_j (i.e. $p(w_i) < p(w_j)$), or (ii) w_j has a precedence dependency on w_i (i.e. $w_i \leftarrow w_j$), or (iii) both items are planned for the same release and have no precedence dependency and w_i has a higher ratio of mean value to mean effort than wj.

From the work sequence, we assign work items to release periods such that the k^{th} item in the work sequence is delivered in release i if, and only if, the cumulative effort to develop the work sequence up to item k is more than the cumulative team capacity up to release period $i-1$ and less or equal to the cumulative team capacity up to release period i:

$$\sum_{j=1}^{i-1} capacity(j) < \sum_{j=1}^{k} effort(ws(j)) \leq \sum_{j=1}^{i} capacity(j) \quad (7)$$

where $ws(j)$ is the j^{th} element in the work sequence ws. This condition ensures that the total effort in each release period does not exceed the team capacity for that period and that work items are released in order of the work sequence derived from the release plan.

4.4 Evaluating Net Present Value

BEARS computes the expected net present value of a release plan p as the mean net present value of the N release scenarios that simulate p.

In a release scenario s, the value of work item w in period i, noted *value(w, s, i)*, equals $value(w)$ if w was released before period i (i.e. $s(w) < i$), or 0 otherwise. The total value delivered during period i of s is the sum of the value delivered by all work items: $value(s,i) = \sum_{w \in WI} value(w)$. The Net Present Value (NPV) of release scenario s is then defined as:

$$NPV(s) = \sum_{i=1}^{L} \frac{value(s,i)}{(1+r)^i} - B \quad (8)$$

where B is the allocated budget for all periods in the planning horizon. If no budget is specified, BEARS uses $B = 0$.

Note that net present value of a release plan p is a random variable whose distribution is approximated by the simulated scenarios' net present values. The expected NPV of a release plan is the mean NPV over that distribution:

$$ENPV(p) = E[NPV(p)] \quad (9)$$

In addition to computing the expected NPV, BEARS can also compute other statistics about NPV such as the loss

probability (the probability that the net present value is negative) and the Value-at-Risk defined as the 5^{th} quantile of the net present value probability distribution.

4.5 Evaluating Expected Punctuality

The punctuality of a release scenario s with respect to a release plan p is defined as the percentage of planned work items delivered on or before their planned release period:

$$Punctuality(s,p) = \frac{\#\{w \in WI \mid s(w) \leq p(w)\}}{\#dom(p)} \quad (10)$$

where *dom(p)* is the domain of p, i.e. the set of work items in the release plan.

The expected punctuality of a release plan p is the mean punctuality of the N release scenarios that simulate p:

$$ExpectedPunctuality(p) = E[Punctuality(s,p)] \quad (11)$$

A release plan with a 90% expected punctuality means that 90% of work items are expected be delivered in their planned release. This expected punctuality metric is motivated by the need for a simple metric to help release planners and software development teams communicate the uncertainty associated with different release plans to clients and other stakeholders. The metric also allows release planners to compare release plans with different expected punctuality and analyse tradeoffs between expected punctuality and expected NPV, as explained in the next section.

4.6 Shortlisting Optimal Release Plans

BEARS shortlists a set of Pareto-optimal solutions that maximize expected net present value (ENPV) and expected punctuality (EP). In BEARS, a release plan p is Pareto-optimal if there is no other release plan p' that outperforms p for both criteria, i.e. such that $ENPV(p') > ENPV(p)$ and $EP(p') > EP(p)$. Conversely, a release plan p is not Pareto-optimal if there is another release plan p' that has either better expected punctuality or better expected net present value.

In general, the number of possible release plans will be too large to compute the exact set of Pareto-optimal solutions. BEARS thus relies on multi-objective evolutionary algorithms (MOEAs) to explore the space of candidate release plans and generate a good approximation of the set of Pareto-optimals release plans. Our implementation uses the following MOEAs: NSGA-II [36], MOCell [37], and SPEA2 [38].

To apply such algorithms, BEARS encodes a release plan p as an array of integers where each element represents the release number $p(w)$ assigned to a work item or zero if the work item is not scheduled in the plan. The size of the array is equal to the number of the work items in the product backlog. The MOEAs start by randomly generating an initial population of 100 release plans, then iteratively evolves the population towards release plans with higher ENPV and EP using genetic selection, crossover and mutation. Our implementation uses integer simulated binary crossover with probability $P_c = 0.9$ and polynomial mutation with probability $P_m = \frac{1}{|WI|}$ where $|WI|$ is the number of work items. We have set the MOEAs to terminate after having explored $25,000$ release plans. On termination, they return the Pareto-optimal release plans in their final population.

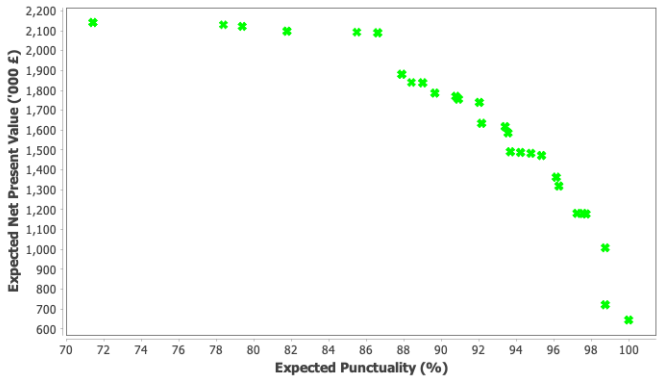


Fig. 3: Expected NPV and punctuality of shortlisted release plans for the local government project.

During mutation and crossover, the MOEAs may generate candidate release plans that violate the precedence constraints between work items. Following an approach used in previous work, our implementation automatically detects and repairs such violations so that the populations only contain valid release plans [39]. If in the future, BEARS is extended to support more dependency relations between work items, such as such as coupling, exclusion and weak precedence (see Section 3.1), the only required change will be to extend the detection and repair of constraints violations during mutation and crossover.

When the search terminates, BEARS displays the shortlisted release plans' Pareto-front. For example, Figure 3 shows the returned Pareto-front for our local government problem. Visualising such Pareto-front helps release planners analyse trade-off between expected value and punctuality. The figure shows that the release plan with the highest ENPV, on the top left, has an expected punctuality around 70%. With only slight decrease in ENPV, release planners could select release plans with expected punctuality of up to 86%. Achieving higher expected punctuality requires further sacrifice in ENPV. Achieving an expected punctuality of 95% requires reducing ENPV to around 1,500K and achieving 100% expected punctuality requires reducing ENPV to around 600K. The figure will thus help release planners and clients make informed decision about their preferred trade-off between expected NPV and punctuality.

5 Evaluation

We have evaluated BEARS in three experiments. The first compares BEARS to EVOLVE-II on our local government release planning problem. The second extends the first by comparing BEARS to more release planning methods on a larger number of release planning problems. The third evaluates the performance of the alternative MOEAs used in BEARS (NSGA-II, SPEA2, and MOCell).

5.1 Illustrative Example: BEARS vs. EVOLVE-II

Our first experiment compares BEARS to EVOLVE-II on our local government example. We have chosen EVOLVE-II

because it epitomizes the use of value points in release planning. It is also the most prominent release planning method in the literature [4], [5], is supported by a professional tool, and has been applied in industrial contexts.

The first objective of this experiment is to provide a a concrete illustrative example of the differences between release plans shortlisted by BEARS and EVOVLE-II. The second objective is to introduce our approach to comparing alternative release planning methods on a same problem. The same approach will be used in our second experiment, and introducing this approach on a single example will make the second experiment easier to explain.

5.1.1 Experiment Design

Section 4 presented the application of BEARS on our running example. The shortlisted release plans were shown in Figure 3.

Imagine now that we had used EVOLVE-II instead of BEARS in the same situation. Starting from the same product backlog described in Section 3.1, we would have estimated the work items' effort and value as described in Section 3.2 and 3.3. This would have resulted in estimates such as those shown Table 2. We would also have had to define the release weights and capacities. We would then have used ReleasePlanner, the tool supporting EVOLVE-II, to shortlist 10 release plans that maximize value points (Eq. 4) subject to capacity constraints (Eq. 1).

Our experiment will simulate this process of applying EVOLVE-II. Instead of eliciting effort and value estimates from stakeholders, we will derive such estimates from the BEARS estimates by assuming that the stakeholders' EVOLVE-II estimates would be consistent with the estimates they have given using BEARS. This allows us to compare the outputs of BEARS and EVOLVE-II optimisation models independently from differences between their effort and value elicitation techniques. For our experiments, we set the EVOLVE-II effort and value point estimates to be to be the mean of the value and effort probability distributions in BEARS, normalised on a scale from 0 to 9. We chose a scale from 0 to 9 because it is the default scale in EVOLVE-II. Likewise, we define other inputs of the EVOLVE-II model so that they are consistent with the corresponding parameters in the BEARS model. We define the team capacity for each release period in EVOLVE-II from the team capacity in the BEARS model using the same approach used to scale effort estimate. We set the weight of each release period i to be $(1+r)^{-i}$ where r is the NPV discount rate in the BEARS model. This ensures that both models apply the same relative weights to sum up values in different release periods.

Once all model inputs have been defined, EVOLVE-II shortlists release plans with the highest value points (Eq. 4) that satisfy the effort capacity constraints (Eq. 1). For this experiment, we have used both ReleasePlanner 2.0, the commercial tool supporting EVOLVE-II [40], and our own implementation of EVOLVE-II. The only difference between the two implementation is the optimization algorithm. ReleasePlanner 2.0 uses an unspecified evolutionary algorithm [3], whereas our implementation uses the same MOEAs with repair as BEARS. In this example, we have used NSGA-II. On this example, our implementation finds

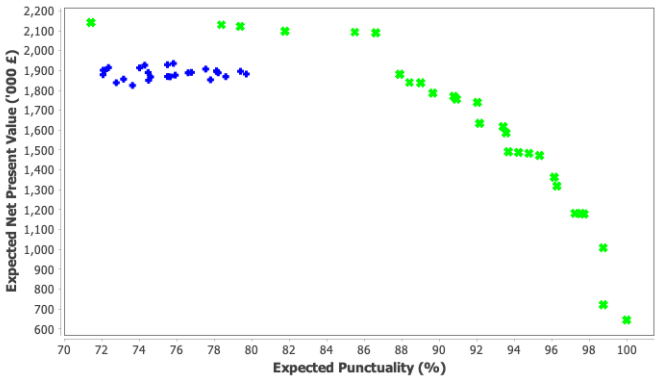


Fig. 4: The BEARS (green crosses) and EVOLVE-II (blue diamonds) shortlists for the local government project.

release plans with higher value points than those found by ReleasePlanner. We have thus decided to compare BEARS against our own implementation of EVOLVE-II because it allows for a fairer comparison that is not affected by the performance of optimisation algorithms.

5.1.2 Results

Figure 4 shows the expected NPV and expected punctuality of release plans in the BEARS and EVOLVE-II shortlists. For the release plans in the EVOLVE-II shortlist, we have used the BEARS model to compute their expected net present value and expected punctuality. For a fair comparison, we include as many release plans in the EVOVLE-II shortlist as they are in the BEARS shorlist (27 in this example)

Figure 4 shows that:

1) The two shortlists are disjoint; none of the release plans shortlisted by EVOLVE-II are shortlisted by BEARS, and vice-versa.
2) The BEARS shortlist *strictly dominates* the EVOLVE-II shortlist; in other words, for every release plan in the EVOLVE-II shortlist there is at least one release plan in the BEARS shortlist that has higher expected NPV and higher expected punctuality.
3) The maximum expected NPV is nearly 10% higher in the BEARS shortlist. The difference is due to EVOLVE-II evaluating value points under the assumption that all work items are delivered on time. Hence, the release plans with the most value points are not necessarily those with the highest expected NPV.
4) The expected punctuality in the BEARS shortlist ranges from 72% to 100%. In contrast, the highest expected punctuality in the EVOLVE-II shortlist is around 80%. Most importantly, in practice EVOLVE-II users would not be aware of the release plans' expected punctuality.

In summmary, Figure 4 shows shows that on this example BEARS shortlists release plans that are better than those shortlisted by EVOLVE-II in terms of expected NPV and expected punctuality. BEARS also provide information about expected punctuality that is missing in EVOLVE-II.

In terms of run-time, BEARS is slower than EVOLVE-II. In this example, BEARS shortlists release plans in 35

seconds, compared to 3 seconds for EVOLVE-II. In our implementation, both methods are configured to stop after evaluating a fixed number of release plans ($25,000$). The run-time increase in BEARS is due to the Monte-Carlo simulation required to evaluate each candidate release plans.

5.1.3 Threats to Validity

External validity. This experiment compares BEARS to EVOLVE-II on a single example. The next section will extend this comparision to more release planning problems.

Internal validity. We have evaluated the release plans shortlisted by EVOVLE-II in terms of the BEARS objectives of maximizing expected NPV and expected punctuality, even though such objectives are not used by EVOLVE-II. Doing so is appropriate because our objective is to study differences between release plans shortlisted by BEARS and EVOLVE-II for the local government release planning problem where two important stakeholders' concerns are the financial implications and punctuality of release plans. The BEARS model provides one way to model these two concerns. The EVOLVE-II model provides another, lighter way to model the financial concern using value point and ignores the punctuality concern due to the method's lack of support for probabilistic reasoning. This experiment showed that using the simpler EVOLVE-II model comes at a cost of shortlisting release plans with lower expected NPV and punctuality.

In this experiment, we have simulated the application of EVOLVE-II by deriving its inputs from the BEARS model instead of applying EVOLVE-II independently from BEARS. This allowed us to minimize the difference between the application of the two methods. If EVOLVE-II had been applied independently from BEARS, it would most likely have led to different effort and value estimates and probably even larger differences between the shortlisted release plans.

We have used EVOLVE-II's default 9 point scale for estimating effort and value. Using a more precise scale might lead to different results. Our next experiment will address this concern by comparing BEARS to a method we call BEARS-*deterministic* where the mean effort and value estimates in BEARS are not scaled down to a 9 point scale.

Finally, this experiment compared a single run of BEARS to a single run of EVOLVE-II. Since both methods use a stochastic optimisation algorithm, different runs may generate different shortlists. Our second experiment will address this concern by repeating each comparison 30 times.

5.2 Comparing BEARS to Other Methods

Our second experiment generalizes the first by comparing BEARS to 4 release planning methods on 32 release planning problems drawn from 8 product backlogs. This experiment aims to answer the following questions:

- Q1 How does BEARS compare to release planning methods that ignore uncertainty?
- Q2 How does BEARS compare to release planing methods under uncertainty with fixed-scope releases?

5.2.1 Experiment Design

(A) Release Planning Methods. Table 3 lists the release planning methods in our experiment.

The first two, EVOLVE-II and BEARS-*deterministic*, are used to answer Q1. EVOVLE-II was introduced in our first experiment. BEARS-*deterministic* is a variant of BEARS where effort and value estimates are single values instead of probability distributions. The optimisation problem is to maximise NPV (Eq. 8) subject to effort capacity constraints (Eq. 1). Like EVOLVE-II, the method shortlists the top 10 release plans. Comparing BEARS to BEARS-*deterministic* allows us to isolate the effect of modelling uncertainty from other differences between BEARS and EVOLVE-II (in particular the 9-point scale used in EVOLVE to estimate story points and value points). For both EVOLVE-II and BEARS-deterministic, our experiment will shortlist the top n release plans, where n is the maximum of the default shortlist size (i.e. 10) and the number of release plans in the BEARS shortlist. This allows for a fairer comparison between shortlists than if we restricted the EVOLVE-II and BEARS-*deterministic* to only 10 release plans.

The other two methods in Table 3, EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* are used to answer Q2. EVOLVE-*with-uncertainty* is based on a previously published variant of EVOLVE that models uncertainty about effort but not value [9]. The method simulates effort uncertainty assuming fixed-scope releases and the optimisation problem has two objectives: maximise value points (Eq. 4) and maximise the probability of delivering all features on time (Eq. 2). BEARS-*fixed-scope* is a variant of BEARS that uses the same uncertain effort and value estimates as BEARS but simulates effort uncertainty assuming fixed-scope releases instead of flexible-scope. Comparing BEARS to BEARS-*fixed-scope* allows us to study the difference between fixed-scope and flexible-scope simulations independently from other factors. The shortlists in EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* are the set of Pareto-optimal solutions returned by the multi-objective optimisation algorithm. Shortlists of different methods can therefore be of different sizes.

In our experiments, we have used our own implementation of each method. As stated in Section 5.1, our implementation of EVOLVE-II uses the sames MOEAs with repair as BEARS but is otherwise equivalent to ReleasePlanner 2.0, the professional tool supporting EVOLVE-II [40]. The variants of EVOLVE that deal with uncertainty have no publicly available implementation; we therefore had to build our own. For consistency, we have tested all methods using NSGA-II as MOEA. This reduces the risk that differences between the methods' shortlists are due to the MOEA rather than due to more fundamental differences in the optimisation models.

(B) Release Planning Problems. Table 4 lists the 8 product backlogs in our experiment. For each product backlog, we consider 4 release planning problems by varying the planning horizon from 2 to 5 periods. The local government project is the illustrative example we have used throughout the paper. The Release Planner, Word Processor, and RALIC product backlogs have been used in previous studies to evaluate the performance of MOEAs on release planning problems [19], [41], [42], [43], [44]. The product backlogs for the Release Planner and Word Processor projects contains candidate features for future releases of the Release Planner tool and for a word processor, respectively. The features

TABLE 3: Release Planning Methods in Our Empirical Evaluation

Method	Effort estimates	Value estimates	Simulation Method	Optimisation problem
EVOLVE II	story points	value points	deterministic	Maximise value points subject to capacity constraint
BEARS-*deterministic*	man-days	economic value	deterministic	Maximise NPV subject to capacity constraint
EVOLVE-*with-uncertainty*	uncertain story points	value points	stochastic, fixed-scope	Maximise value points; Maximise on-time probability
BEARS-*fixed-scope*	uncertain man-days	uncertain economic value	stochastic, fixed-scope	Maximise expected NPV; Maximise on-time probability
BEARS	uncertain man-days	uncertain economic value	stochastic, flexible-scope	Maximise expected NPV; Maximise expected punctuality

TABLE 4: Release Planning Problems in Our Empirical Evaluation

Release Planning Problem	Backlog Size	Number of Constraints	Original effort estimates	Original value estimates
Local Government Project	20	23	uncertain man-days	uncertain economic value
Release Planner	25	11	man-hours	value points
Word Processor	50	65	man-hours	value points
RALIC	143	0	man-hours	value points
Synthetic-30	30	11	uncertain man-days	uncertain economic value
Synthetic-50	50	15	uncertain man-days	uncertain economic value
Synthetic-100	100	30	uncertain man-days	uncertain economic value
Synthetic-200	200	51	uncertain man-days	uncertain economic value

were elicited and evaluated during exploratory studies for a variant of the EVOLVE method [45]. The product backlog for the RALIC project includes requirements for a future building access control system at University College London. The requirements were elicited from and evaluated by 87 stakeholders using an online tool that leveraged stakeholders' relationships to drive the requirements elicitation and prioritization process [46]. The last 4 product backlogs are synthetic backlogs of size 30, 50, 100, and 200, respectively. The work items precedence constraints, effort and value estimations have been generated at random. These synthetic models will allow us to observe how BEARS performance varies as the number of work items in the product backlog increases.

For the Release Planner, Word Processor and RALIC product backlogs, the original effort and value estimates do not include uncertainty. To be able to apply BEARS on these problems, we have artificially added uncertainty to the estimates by simulating the uncertainty elicitation process described in Section 4.2. For an original point-based effort estimate x, we have set the upper and lower bounds to x and $1.8x$ and the lower quartile, median and upper quartile to $1.2x$, $1.5x$ and $1.7x$. This simulates a situation where the original effort estimate is the most optimistic value and the true development time could take up to 1.8 times this optimistic value. This situation is consistent with studies of software estimations that show that people tend to underestimate development time [22]. For an original value point estimate y, we have set the upper and lower bounds to 0 and $1000y$ and the lower quartile, median and upper quartile to $200y$, $500y$, and $750y$. This simulates a situation where one value point is worth $£1,000$ and the original value estimate is the most optimistic value and the true value could be as low as zero. This is consistent with observations of software project where initial prediction of business value tend to be overestimated [47], [48] We make no claim that the values we have chosen represent the true uncertainty that would have been elicited from real developers and stakeholders in these projects. Since our objective is to compare the outputs of different release planning method when applied on the same inputs, it is sufficient that these inputs are realistic and used consistently across our experiments.

For converting BEARS probability distributions to EVOLVE estimates, we used the approach described in Section 5.1. For the Release Planner, Word Processor, and RALIC product backlogs, we first generate BEARS probability distribution as described in the previous paragraph, then generate new EVOLVE estimates that are proportional to the mean of the BEARS distributions as described in Section 5.1. When applying BEARS-*deterministic*, we use the mean of the BEARS probability distribution as single-point estimate for effort and value.

(C) Evaluation Metrics. A first evaluation metric is to observe how often the BEARS shortlist *strictly dominates* the shortlists of other methods. In our context, a release plan $p1$ strictly dominates a release plan $p2$ if $ENPV(p1) > ENPV(p2)$ and $EP(p1) > EP(p2)$. A shortlist $L1$ strictly dominates a shortlist $L2$ if every release plan in $L2$ is strictly dominated by at least one release plan in $L1$. Strict dominance is the strongest possible form of dominance relation between sets of solutions in multi-objective optimisation problems [49]. If a single release plan in $L2$ is equal to, or is not dominated by some release plan in $L1$, then $L1$ does not strictly dominates $L2$. As in the first experiment, we have used the BEARS model (Equations 9 and 11) to compute the expected NPV and expected punctuality of release plans shortlisted by other methods.

Analysing strict dominance can tell us whether the release plans shortlisted by BEARS are better than those shortlisted by other methods but it does not tell us *how much* better. Our second evaluation metric measures the improvement of a BEARS shortlist compared to the shortlist of another method by comparing their hypervolumes. In multi-objective optimisation problems, the hypervolume of a solution set A, noted *HV(A)* is the volume of the objective space dominated by A [50]. For example, in Figure 4 the hypervolume of a shortlist is the area dominated by the release plans in the shortlist, bounded below by the x-axis (*ENPV*= 0) and to the left by the y-axis (*EP*= 0). In this example, the HV of the BEARS and EVOLVE-II shortlists are 2047 and 1541, respectively. The HV of the BEARS shortlist is larger because it covers a larger area. The Hyper-

TABLE 5: How often the BEARS shortlist dominates the shortlists of other methods and range of hypervolume improvement ratios (HVIR).

BEARS *vs.*	Strictly Dominates	HVIR Mean	HVIR Min-Max
EVOLVE-II	96%	1.17	1.03-1.39
BEARS-*deterministic*	97%	1.18	1.03-1.42
EVOLVE-*with-uncertainty*	79%	1.11	1.01-1.29
BEARS-*fixed-scope*	89%	1.12	1.01-1.32

volume improvement ratio (*HVIR*) of a solution set A over a solution set B is $HV(A)/HV(B)$. In Figure 4, the HVIR of the BEARS shortlist over the EVOLVE-II shortlist is 2047/1541 = 1.33. Hypervolume is a widely used metric to compare solution sets of multi-objective optimisation problems. We have chosen this metric because it has as simple visual interpretation and is compatible with strict dominance, i.e. if A strictly dominates B then $HV(A)/HV(B) > 1$ [49].

(D) Experimental Set Up. The MOEAs used by the release planning methods in Table 3 are stochastic algorithms that can generate different results each time they are executed on a given problem. To account for such randomness, we have executed 30 independent runs of each of our 5 release planning methods on all 32 release planning problems. Each release planning method is thus executed 960 times (30×32). In total, our experiment includes $4,800$ independent runs (5×960), resulting in as many shortlists. All runs were executed on a single PC with Intel Core i5 CPU at 3.20GHz x 4 and 8GB of RAM.

5.2.2 Results

Table 5 summarizes the results of our experiments. Detailed data showing strict dominance and HVIR on each problem individually can be found in Appendix A.

Q1) How does BEARS *compare to release planning methods that ignore uncertainty?* Table 5 shows that out of 968 runs, the BEARS shortlist strictly dominates the EVOLVE-II and BEARS-*deterministic* shortlists in 96% and 97% of the runs, respectively. Furthermore, the BEARS shortlist has an hypervolume that is on average 17% and 18% higher than the EVOLVE-II and BEARS-*deterministic* shortlists, respectively. The hypervolume improvements range from 3% to 42%.

> Analysing uncertainty using BEARS leads to shortlisting release plans with higher expected NPV and expected punctuality than methods that ignore uncertainty. In our experiments, the improvement in hypervolume range from 3% to 42%, with an average of 17%.

Q2) How does BEARS *compare to release planing methods under uncertainty with fixed-scope releases?* Table 5 shows that out of 968 runs, the BEARS shortlist strictly dominates the EVOLVE-*with-uncertainty* and and BEARS-*fixed-scope* shortlists in 79% and 89% of the runs, respectively. Furthermore, the BEARS shortlist has an hypervolume that is on average 11% and 12% higher than the EVOLVE-*with-uncertainty* and and BEARS-*fixed-scope* shortlists, respectively. The hypervolume improvements range from 1% to 32%.

> In the context of fixed-time releases, analysing uncertainty using BEARS leads to shortlisting release plans with higher expected NPV and expected punctuality than methods designed for fixed-scope releases. In our experiments, the improvement in hypervolume range from 1% to 32%, with an average of 11%.

We have also compared the methods' run-times. The average run-time of each method on each release planning problem is reported in Appendix A. The results show that BEARS is on average 25 times slower than the two methods that ignore uncertainty and 5 times slower than the two methods that analyse uncertainty assuming fixed-scope releases. For our largest product backlog of 200 work items and $H = 5$, BEARS takes 10 minutes to run. Although BEARS is significantly slower than other methods, its run time remains acceptable and does not prevent BEARS being used during release planning workshops. In such workshops, release plans are typically shortlisted only once after all work items effort and values estimates have been elicited.

5.2.3 Threats to Validity

External validity. Our results may not generalise beyond the 32 release planning problems considered in this experiment. Release planing problems where work items have different uncertainty or where other model parameters, such as team capacity, have different values could lead to different results. Further experiments on other problems are needed to refine the findings. Nevertheless, this experiment showed important differences can exist between the outputs of different release planning methods, and we have found important differences between BEARS and methods that ignore uncertainty in all release planning problems studied so far.

Internal validity. As in the first experiment, we used the BEARS objectives of expected NPV and expected punctuality to evaluate release plans generated by other methods that optimise different objectives. This evaluation is appropriate because the experiments' objective is to compare different release planning methods in the context for which BEARSis designed, which is the context most commonly found in practice. Also as in the first experiment, we have deliberately aimed to minimize differences in effort and value elicitation techniques. Taking such differences into account is likely to amplify the differences found in the experiments.

Out of scope. Our experiments focussed comparing BEARS shortlists against those of other methods but have ignored other factors that are important in practice. In particular, we have not evaluated the uncertainty elicitation method used in BEARS (Section 4.2), the ability of release planners to understand BEARS' outputs, the overall cost of applying the method, and the general perceived benefits and limitations of the method by release planners, development teams and other stakeholders.

5.3 Evaluating BEARS Optimisation Algorithms

Our third experiment evaluates the performance of BEARS MOEAs. It aims to answer the following questions:

Q3 Do the three MOEAs used in BEARS perform better than a random search?
Q4 Do the three MOEAs used in BEARS have important differences in performance?

Our objective here is not to propose novel MOEAs, nor to look for the best possible algorithms for solving BEARS optimisation problems. This experiment is a sanity check aimed at evaluating whether existing MOEAs are suitable for solving BEARS optimisation problems. Developing and evaluating more efficient algorithms for solving BEARS optimisation problems is left as future work.

5.3.1 Experiment Design

The three MOEAs used in BEARS are NSGA-II, SPEA2, and MOCell (see Section 4.6). We selected these MOEAs because they have readily available implementations in JMetal, the optimisation framework used in BEARS [32]. We compare these algorithms against a random search augmented with the same constraints violation detection and repair technique used with the MOEAs (see Section 4.6). The random search is configured to evaluate the same number of valid release plans as the MOEAs ($25,000$).

We have executed each of these 4 algorithms 30 times on the same 32 release planning problems used in our second experiment. This makes up a total of $3,840$ runs.

For each run, we retrieve the generated shortlist and measure its Hypervolume (HV) [50] and modified Inverted Generational Distance (IGD+) [51].

In this experiment, we compute HV using *normalised* expected NPV values for each problem, where the minimum is 0 and maximum is the highest expected NPV found in all evaluated release plans for that problem. The HV for all problems are thus all measured on the same scale. Better shortlists have *higher* HV.

The IGD+ of a solution set A is the average distance between the true Pareto-optimal solutions and the region in the objective space dominated by A [51]. When, as in our case, the true Pareto optimal solutions are unknown, these solutions are approximated by so-called *reference* Pareto-optimal solutions, which are the non-dominated solutions in the union of all solutions explored by all algorithms over all of independent runs. Better shortlists have *lower* IGD+.

We have chosen HV and IGD+ as our evaluation metrics based on guidance from Li et al. [52] that recommend using these metrics in situations, like ours, where the decision makers' preferences about the qualities of solutions sets are unknown These metrics are recommended because they are compatible with the strict dominance relation, and they cover all typical qualities desired of solutions sets, i.e. their convergence (how close the solutions set is to the true Pareto front), diversity (the extent to which the solutions set includes diverse solutions), and cardinality (the number of solutions in the set).

5.3.2 Results

This section presents the conclusion of this experiment. Detailed data can be found in Appendix B.

Q3) Do the three MOEAs used in BEARS *perform better than a random search?* The experiment show that the three MOEAs perform better than random search and that the differences in HV and IGD+ are statistically significant. In terms of IGD+, the three MOEAS outperform random search in all 32 problems. In terms of HV, the three MOEAS outperform random search in 27 of the 32 problems. In the remaining 5 problems, random search performs better than MOCEll in 4 cases and better than SPEA2 in 1 case. Note that the random search in BEARS uses the same constraint violation detection and repair technique used by the MOEAs. Random search in BEARS is therefore more than a blind search, which may explain why in a few cases it performs better than one of the three MOEA on one of the evaluation criteria, even if overall the MOEAs have better IGD+ and HV.

Q4) Do the three MOEAs used in BEARS *have important differences in performance?* The experiment show no statistically significant differences between SPEA2 and MOCell, and a slight advantage of these two algorithms over NSGA-II. The experiment therefore suggest that SPEA2 and MOCell may find slightly better shortlists than NSGA-II for BEARS optimisation problems.

5.3.3 Threats to Validity

Our third experiment follows common guidelines and practices for evaluating the performance of stochastic multi-objective algorithms [53], [52]. This experiment is therefore subject to the usual limitations of such evaluations, notably the extent to which results can be generalized beyond the 32 release planning problems.

6 Related Work

In Section 3, we saw that most release planning methods ignore uncertainty [4], [5] and that the few methods that analyse uncertainty assume fixed-scope releases [9], [10], [11], [12], [13], which is inconsistent with current industrial practices [8]. In contrast, BEARS is a release planning method under uncertainty that supports fixed-date, flexible-scope release cycles most commonly used in industry.

Previous empirical studies of release planning methods have either focussed on evaluating release planning methods in industry [6], [7], [8], or on comparing the performance of alternative optimisation techniques as in our third experiment. A recent paper references 38 different evaluation of MOEAs for software release planning and presents the most comprehensive such evaluation to date [44]. Like in our third experiment, such evaluation compare the outputs of different MOEAs in the context of a single release planning method. In contrast, our first and second experiments focussed on comparing alternative release planning methods. These experiments enabled us to observe important differences in the release plans shortlisted by BEARS compared to other methods.

The paper's contributions with respect to previous work on release planning are:

1) a novel model for simulating flexible-scope release plans under uncertainty (Section4.3);
2) a novel metric for measuring the expected punctuality of release plans (Section 4.5);
3) an automated tool supporting the optimisation of release plans in BEARS (Section 4.6);

4) an empirical study comparing BEARS to release planning methods that ignores uncertainty (Section 5.2, Q1);
5) an empirical study comparing BEARS to release planning methods under uncertainty that assume fixed-scope releases (Section 5.2, Q2).

The Bayesian approach used in BEARS is similar to one used to support requirements and architecture decisions under uncertainty [33], [54]. The decision models supporting such requirements and architecture decisions are based on quantitative goal models [55], [56], [57]. They allow a richer, more detailed modelling of stakeholders' goals and business value than BEARS, but they do not support reasoning about multiple iterations as done in BEARS. BEARS introduces novel simulation and optimisation models that are specific to release planning.

7 Conclusion

BEARS is the first release planning method that supports the industry practice of fixed-date, flexible-scope releases. We compared BEARS to release planning methods that ignore uncertainty and to previous methods that deal with uncertainty assuming fixed-scope releases.

The paper's main contribution has been to show that analysing uncertainty leads to shortlisting better release plans than if uncertainty is ignored: analysing uncertainty gives release planners more credible information about candidate release plans and leads to identifying release plans with higher expected net present value and punctuality. The paper also shows that optimal release plans for flexible-date, fixed-scope releases are not optimal in the context of fixed-date, flexible-scope releases.

As future work, we plan on extending BEARS to analyse a wider range of criteria and work items during release planning such as: analysing fairness and multiple value criteria evaluated from multiple perspectives [26]; managing technical debt during release planing [58]; supporting decisions about what experiments to perform (e.g. as A/B tests) and what data to collect to support future release planning decisions; and analysing objective data (e.g. from the software development process, software usage, and users' feedback) to update uncertainty about feature's effort and value so as to inform future decisions [59].

References

[1] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.
[2] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.
[3] G. Ruhe, *Product Release Planning: Methods, Tools and Applications*. CRC Press, 2010.
[4] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique, "A systematic review on strategic release planning models," *Information and software technology*, vol. 52, no. 3, pp. 237–248, 2010.
[5] D. Ameller, C. Farré, X. Franch, and G. Rufian, "A survey on software release planning models," in *Proceedings of the 17th International Conference on Product-Focused Software Process Improvement (PROFES 2016)*. Springer, 2016, pp. 48–65.
[6] Amandeep, G. Ruhe, and M. Stanford, *Intelligent Support for Software Release Planning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 248–262.
[7] J. Momoh and G. Ruhe, "Release planning process improvement an industrial case study," *Software Process: Improvement and Practice*, vol. 11, no. 3, pp. 295–307, 2006.
[8] M. Lindgren, R. Land, C. Norström, and A. Wall, "Key aspects of software release planning in industry," in *19th Australian Conference on Software Engineering*. IEEE, 2008, pp. 320–329.
[9] G. Ruhe and D. Greer, "Quantitative studies in software release planning under risk and resource constraints," in *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE '03)*. IEEE Computer Society, 2003.
[10] K. McDaid, D. Greer, F. Keenan, P. Prior, G. Coleman, and P. S. Taylor, "Managing uncertainty in agile release planning." in *SEKE*, 2006, pp. 138–143.
[11] K. Logue and K. McDaid, "Agile release planning: Dealing with uncertainty in development time and business value," in *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008)*. IEEE, 2008, pp. 437–442.
[12] B. P. Barbosa, E. A. Schmitz, and A. J. Alencar, "Generating software-project investment policies in an uncertain environment," in *Systems and Information Engineering Design Symposium (SIEDS 2008)*. IEEE, 2008, pp. 178–183.
[13] O. Oni and E. Letier, "Optimizing the incremental delivery of software features under uncertainty," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'16)*. Springer International Publishing, 2016, pp. 36–41.
[14] T. DeMarco and T. Lister, *Waltzing with Bears: Managing Risk on Software Projects*. New York, NY, USA: Dorset House Publishing Co., Inc., 2003.
[15] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Information and software technology*, vol. 43, no. 14, pp. 883–890, 2001.
[16] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and software technology*, vol. 46, no. 4, pp. 243–253, 2004.
[17] M. Denne and J. Cleland-Huang, *Software by Numbers: Strategies for High Return, Low Risk Application Development*. Pearson Education, 2003.
[18] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.
[19] Y. Zhang, M. Harman, and S. L. Lim, "Empirical evaluation of search based requirements interaction management," *Information and Software Technology*, vol. 55, no. 1, pp. 126–152, 2013.
[20] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.
[21] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on software engineering*, vol. 33, no. 1, 2007.
[22] M. Jørgensen, "What we do and don't know about software development effort estimation," *IEEE software*, vol. 31, no. 2, pp. 37–40, 2014.
[23] L. Li, M. Harman, E. Letier, and Y. Zhang, "Robust next release problem: handling uncertainty during optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1247–1254.
[24] L. Li, M. Harman, F. Wu, and Y. Zhang, "The value of exact analysis in requirements selection," *IEEE Transactions on Software Engineering*, pp. 1–1, 2016.
[25] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, *Value-based software engineering*. Springer Science & Business Media, 2006.
[26] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, "Fairness analysis in requirements assignments," in *16th IEEE International Requirements Engineering Conference (RE'08)*. IEEE, 2008, pp. 115–124.
[27] S. Tockey, *Return on software*. Addison-Wesley, 2005.
[28] M. Cantor, "Calculating and improving ROI in software and system programs," *Commun. ACM*, vol. 54, no. 9, pp. 121–130, Sep. 2011.
[29] A. J. Alencar, C. A. Franco, E. A. Schmitz, and A. L. Correa, "A statistical approach for the maximization of the financial benefits yielded by a large set of MMFs and AEs," *Computing and Informatics*, vol. 32, no. 6, pp. 1147–1169, 2014.
[30] E. Mattos, M. Vieira, E. A. Schmitz, and A. J. Alencar, "Applying game theory to the incremental funding method in software projects," *Journal of Software*, vol. 9, no. 6, pp. 14–35, 2014.
[31] D. Hubbard, *How to Measure Anything: Finding the Value of Intangibles in Business*. Wiley, 2010.

[32] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jMetal multi-objective optimization framework," in *Proc. of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2015, pp. 1093–1100.

[33] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *36th International Conference on Software Engineering*, 2014, pp. 883–894.

[34] A. O'Hagan, C. Buck, A. Daneshkhah, J. Eiser, P. Garthwaite, D. Jenkinson, J. Oakley, and T. Rakow, *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, 2006.

[35] D. E. Morris, J. E. Oakley, and J. A. Crowe, "A web-based tool for eliciting probability distributions from experts," *Environmental modelling & software*, vol. 52, pp. 1–4, 2014.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[37] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "MOCell: A cellular genetic algorithm for multiobjective optimization," *Int. J. Intell. Syst.*, vol. 24, no. 7, pp. 726–746, Jul. 2009.

[38] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," 2001.

[39] J. Sagrado, I. M. Águila, and F. J. Orellana, "Multi-objective ant colony optimization for requirements selection," *Empirical Softw. Engg.*, vol. 20, no. 3, pp. 577–610, Jun. 2015.

[40] Expert Decisions Inc. (2017) Expert decisions' home page. https://www.expertdecisions.com.

[41] A. M. Pitangueira, P. Tonella, A. Susi, R. S. P. Maciel, and M. Barros, "Minimizing the stakeholder dissatisfaction risk in requirement selection for next release planning," *Information and Software Technology*, vol. 87, pp. 104–118, 2017.

[42] A. A. Araújo, M. Paixao, I. Yeltsin, A. Dantas, and J. Souza, "An architecture based on interactive optimization and machine learning applied to the next release problem," *Automated Software Engineering*, vol. 24, no. 3, pp. 623–671, 2017.

[43] Y. Zhang, E. Alba, J. J. Durillo, S. Eldh, and M. Harman, "Today/future importance analysis," in *12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 1357–1364.

[44] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, and S. Brinkkemper, "An empirical study of meta-and hyper-heuristic search for multi-objective release planning," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 1, p. 3, 2018.

[45] M. R. Karim and G. Ruhe, "Bi-objective genetic search for release planning in support of themes," in *International Symposium on Search Based Software Engineering*. Springer, 2014, pp. 123–137.

[46] S. L. Lim and A. Finkelstein, "StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE transactions on software engineering*, vol. 38, no. 3, pp. 707–735, 2012.

[47] National Audit Office, "Over-optimism in government projects," 2013.

[48] B. Flyvbjerg and A. Budzier, "Why your IT project may be riskier than you think," *Harvard Business Review*, vol. 89, no. 9, pp. 23–25, 2011.

[49] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on evolutionary computation*, vol. 7, no. 2, pp. 117–132, 2003.

[50] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[51] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 110–125.

[52] M. Li, T. Chen, and X. Yao, "A critical review of: a practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering: essay on quality indicator selection for SBSE," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*. ACM, 2018, pp. 17–20.

[53] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 1–10.

[54] S. A. Busari and E. Letier, "RADAR: A lightweight tool for requirements and architecture decision analysis," in *39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 552–562.

[55] A. van Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*. Wiley, 2009.

[56] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *12th International Symposium on the Foundation of Software Engineering (FSE 2004)*, vol. 29, no. 6. ACM, 2004, pp. 53–62.

[57] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in *19th IEEE International Requirements Engineering Conference*. IEEE, 2011, pp. 79–88.

[58] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt: A systematic literature review," *Information and Software Technology*, vol. 64, pp. 52–73, 2015.

[59] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

Olawole Oni Olawole Oni received the BSc degree in computer science from University of Ilorin, Nigeria and the MSc degree in Software Engineering from University College London (UCL). He is a PhD student at UCL researching software release planning under uncertainty.

Emmanuel Letier Emmanuel Letier is an Associate Professor in the Department of Computer Science at University College London where he heads to Software Systems Engineering Research group. His research is in requirements engineering and software architecture.

APPENDIX A
RESULTS FOR EXPERIMENT IN SECTION 5.2

This appendix presents the detailed results of the experiment comparing the shortlists generated by BEARS and other methods (Section 5.2).

A.1 Strict Dominance

Table 6 reports how often bears generates a shortlist that strictly dominates the shortlist of other methods when each method is executed 30 times. In answer to Q1, the table shows that BEARS strictly dominates EVOLVE-II and BEARS-*deterministic* in 95% and 99% of the runs, respectively. In answer to Q2, the table shows that BEARS strictly dominates EVOLVE-with-uncertainty and BEARS-*fixed-scope* in 77% and 89% of the runs, respectively.

A.2 Hypervolume Improvement Ratios

Table 7 reports the mean, minimum and maximum hypervolume improvement ratios over 30 runs for each release planning problem. In answer to Q1, the table shows that BEARS generates shortlists whose hypervolumes are on average 30% and 36% higher than those generated by EVOLVE-II and BEARS-*deterministic*, respectively. In answer to Q2, the table shows that BEARS generates shortlists whose hypervolumes are on average 6% and 11% higher than those generated by EVOLVE-*with-uncertainty* and BEARS-*fixed-scope*, respectively.

We have checked that all observed differences in hypervolumes between BEARS and other methods are statistically significant using Mann-Whitney U test with $p < .05$. All p-values are reported in Table 8.

A.3 Illustrative Examples of Generated Shortlists

To help us understand the practical significance of the above results in specific contexts, Figures 5 and 6 show examples of shortlists generated by all release planning methods for each product backlog and a planning horizon $H = 3$. These examples all correspond to the first of the 30 runs of each method.

For each release planning problem, the figure shows how much the shortlist generated by BEARS dominates the shortlists generated by the other methods. In Figure 5, we observe that the dominance of BEARS over methods that ignore uncertainty is important in practice. For each of these examples, we can make similar observations to those made in our first experiment (Section 5.1). In Figure 6, we see that the difference between BEARS and methods that analyse uncertainty assuming fixed-scope releases is smaller but still important enough to justify using BEARS for projects with fixed-date, flexible-scope release cycles.

A.4 Run-times

Table 9 shows the average run-time of each method for each release planning problem.

We observe that BEARS is much slower than methods that ignore uncertainty (roughly between 10 to 50 times slower depending on the problem) and slower that methods that analyse uncertainty under the assumption of fixed-scope releases (between 2 and 8 times slower). On the largest product backlog with 200 items and a planning horizon $H = 5$, BEARS takes nearly 10 minutes to generate a shortlist. This time is not ideal but acceptable for BEARS's intended use during release planning workshops. For the local government project, BEARS returns the shortlist in about 1 minute.

APPENDIX B
RESULTS FOR EXPERIMENT IN SECTION 5.3

Table 10 reports the median HV and IGD+ of each algorithm over 30 runs for each release planning problem. In each row, the best result is highlighted in a light grey and the worst in dark grey.

We have used Mann-Whitney U test to evaluate whether the observed differences in HV and IGD+ between algorithms are statistically significant. Table 11 reports the results of these tests.

In response to Q3, Table 10 shows that overall all three methods outperfrom random-search as discussed in the paper.

In reponse to Q4, Table 11 shows that overall the difference between SPEA2 and MOCell is not statistically significant (last two rows of the last column). Table 10 and Table 11 show a slight advantage of SPEA2 and MOCell over NSGA-II.

TABLE 6: Proportion of runs where the BEARS shortlists strictly dominates the shortlist of other methods.

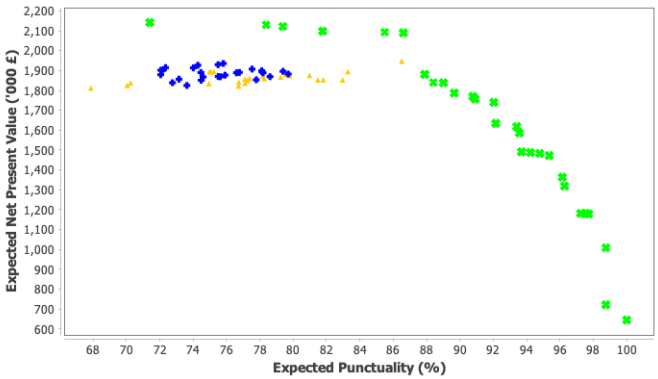
		BEARS *vs.*			
Product Backlog	H	EVOLVE-II	Bears-deterministic	Evolve-with-uncertainty	Bears-fixed-scope
Local Government Project	2	30/30	21/30	18/30	18/30
	3	30/30	30/30	19/30	21/30
	4	21/30	27/30	22/30	23/30
	5	27/30	29/30	21/30	27/30
Release Planner	2	23/30	25/30	25/30	21/30
	3	30/30	30/30	30/30	28/30
	4	30/30	30/30	30/30	30/30
	5	30/30	30/30	30/30	30/30
Word Processor	2	23/30	24/30	20/30	26/30
	3	29/30	29/30	27/30	23/30
	4	30/30	30/30	30/30	30/30
	5	30/30	27/30	26/30	30/30
RALIC	2	30/30	30/30	30/30	30/30
	3	29/30	30/30	27/30	29/30
	4	24/30	30/30	30/30	29/30
	5	22/30	30/30	20/30	27/30
Synthetic-30	2	30/30	30/30	18/30	20/30
	3	30/30	29/30	20/30	30/30
	4	30/30	30/30	21/30	30/30
	5	30/30	30/30	27/30	30/30
Synthetic-50	2	30/30	29/30	20/30	20/30
	3	30/30	30/30	20/30	21/30
	4	30/30	30/30	18/30	24/30
	5	30/30	30/30	23/30	28/30
Synthetic-100	2	30/30	30/30	21/30	28/30
	3	30/30	30/30	24/30	29/30
	4	30/30	30/30	20/30	28/30
	5	30/30	30/30	20/30	29/30
Synthetic-200	2	30/30	30/30	20/30	27/30
	3	30/30	30/30	22/30	30/30
	4	30/30	30/30	20/30	29/30
	5	30/30	30/30	25/30	30/30
Overall		918/960 (96%)	930/960 (97%)	754/960 (79%)	855/960 (89%)

TABLE 7: Hypervolume Improvement Ratios of BEARS with respect to other release planning methods.

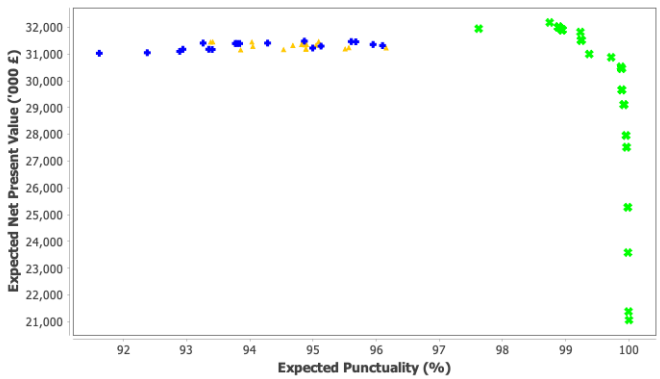
		BEARS *vs.*							
		EVOLVE-II		Bears-deterministic		Evolve-with-uncertainty		Bears-fixed-scope	
Product Backlog	H	Mean	Range	Mean	Range	Mean	Range	Mean	Range
Local Government Project	2	1.22	1.19 –1.26	1.27	1.22 –1.34	1.06	1.03 –1.07	1.05	1.00 –1.12
	3	1.24	1.20 –1.33	1.30	1.22 –1.55	1.12	1.07 –1.23	1.12	1.05 –1.33
	4	1.28	1.16 –1.76	1.32	1.19 –1.51	1.11	1.07 –1.17	1.15	1.10 –1.27
	5	1.34	1.18 –1.48	1.47	1.30 –1.68	1.11	1.05 –1.19	1.16	1.11 –1.24
Release Planner	2	1.05	0.97 –1.10	1.06	1.02 –1.11	1.02	0.95 –1.07	1.01	0.93 –1.06
	3	1.07	1.05 –1.10	1.08	1.05 –1.11	1.06	1.03 –1.09	1.05	1.01 –1.08
	4	1.06	1.04 –1.09	1.06	1.04 –1.09	1.05	1.03 –1.08	1.06	1.03 –1.11
	5	1.07	1.06 –1.10	1.09	1.06 –1.15	1.06	1.03 –1.08	1.08	1.05 –1.12
Word Processor	2	1.00	0.95 –1.05	1.01	0.96 –1.05	1.00	0.94 –1.05	1.01	0.94 –1.03
	3	1.04	1.02 –1.06	1.04	1.02 –1.06	1.03	1.00 –1.05	1.03	1.00 –1.05
	4	1.06	1.05 –1.07	1.05	1.04 –1.06	1.05	1.04 –1.05	1.03	1.02 –1.04
	5	1.04	1.01 –1.04	1.03	1.00 –1.04	1.02	0.99 –1.03	1.04	1.01 –1.07
RALIC	2	1.10	1.04 –1.20	1.12	1.08 –1.16	1.07	1.03 –1.12	1.08	1.04 –1.14
	3	1.08	1.02 –1.15	1.14	1.08 –1.22	1.06	0.98 –1.12	1.08	1.02 –1.15
	4	1.09	1.02 –1.14	1.16	1.11 –1.25	1.07	1.01 –1.11	1.08	1.03 –1.15
	5	1.08	1.01 –1.15	1.16	1.07 –1.26	1.06	1.00 –1.09	1.08	1.02 –1.17
Synthetic-30	2	1.15	1.09 –1.67	1.11	1.05 –1.25	1.03	1.01 –1.09	1.02	1.00 –1.05
	3	1.14	1.09 –1.20	1.11	1.06 –1.20	1.08	1.03 –1.11	1.08	1.04 –1.13
	4	1.16	1.09 –1.23	1.15	1.10 –1.30	1.11	1.06 –1.15	1.12	1.06 –1.17
	5	1.16	1.12 –1.27	1.16	1.11 –1.22	1.14	1.11 –1.18	1.15	1.11 –1.23
Synthetic-50	2	1.15	1.07 –1.30	1.14	1.03 –1.26	1.05	1.04 –1.08	1.02	1.00 –1.06
	3	1.23	1.10 –1.36	1.25	1.12 –1.42	1.08	1.05 –1.12	1.08	1.03 –1.12
	4	1.28	1.16 –1.42	1.30	1.22 –1.45	1.13	1.05 –1.25	1.14	1.08 –1.20
	5	1.30	1.20 –1.44	1.30	1.21 –1.40	1.17	1.10 –1.26	1.20	1.14 –1.27
Synthetic-100	2	1.09	1.04 –1.14	1.08	1.05 –1.15	1.07	1.04 –1.12	1.09	1.04 –1.13
	3	1.21	1.14 –1.31	1.20	1.10 –1.33	1.15	1.09 –1.22	1.16	1.10 –1.26
	4	1.32	1.21 –1.44	1.32	1.20 –1.43	1.20	1.09 –1.32	1.22	1.12 –1.32
	5	1.39	1.28 –1.52	1.38	1.30 –1.51	1.25	1.16 –1.34	1.26	1.18 –1.35
Synthetic-200	2	1.12	1.06 –1.22	1.11	1.05 –1.20	1.15	1.07 –1.23	1.16	1.09 –1.25
	3	1.17	1.10 –1.24	1.19	1.11 –1.26	1.21	1.12 –1.29	1.24	1.13 –1.31
	4	1.28	1.21 –1.38	1.26	1.18 –1.34	1.32	1.17 –1.92	1.33	1.17 –1.55
	5	1.38	1.24 –1.50	1.40	1.23 –1.50	1.37	1.25 –1.69	1.36	1.18 –1.54
Overall		1.17	1.03 –1.39	1.18	1.03 –1.42	1.11	1.01 –1.29	1.12	1.01 –1.32

TABLE 8: Statistical Significance (p-value) of the observed difference in hypervolume between BEARS and other methods over 30 runs using Mann-Whitney U test.

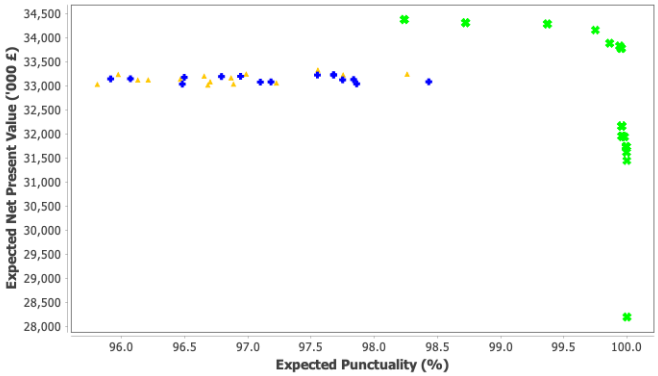
		BEARS *vs.*			
Product Backlog	H	EVOLVE-II	Bears-deterministic	Evolve-with-uncertainty	Bears-fixed-scope
Local Government Project	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$5.77E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Release Planner	2	$2.11E-07$	$2.10E-08$	$4.58E-04$	$1.25E-02$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.51E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Word Processor	2	$3.01E-01$	$6.04E-02$	$4.87E-01$	$3.85E-02$
	3	$2.87E-11$	$6.37E-11$	$1.15E-10$	$3.06E-09$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.74E-10$	$5.84E-10$	$2.87E-11$
RALIC	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$6.37E-11$	$2.87E-11$	$3.31E-10$	$2.05E-10$
	4	$3.18E-11$	$2.87E-11$	$3.18E-11$	$2.87E-11$
	5	$3.51E-11$	$2.87E-11$	$9.44E-11$	$3.51E-11$
Synthetic-30	2	$2.87E-11$	$2.87E-11$	$4.73E-11$	$6.41E-10$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-50	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.39E-08$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-100	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-200	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Overall		$5.43E-116$	$3.21E-149$	$1.36E-104$	$2.71E-114$



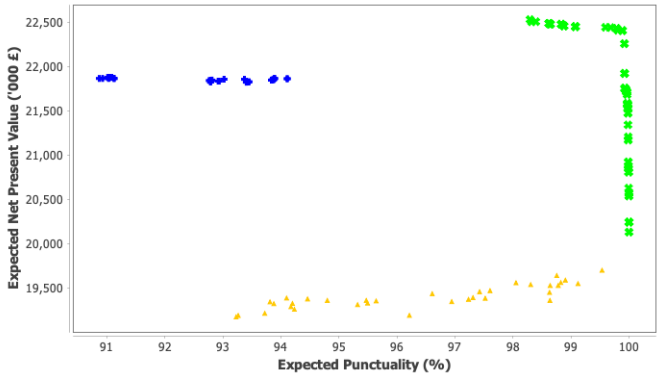
(a) Local Government



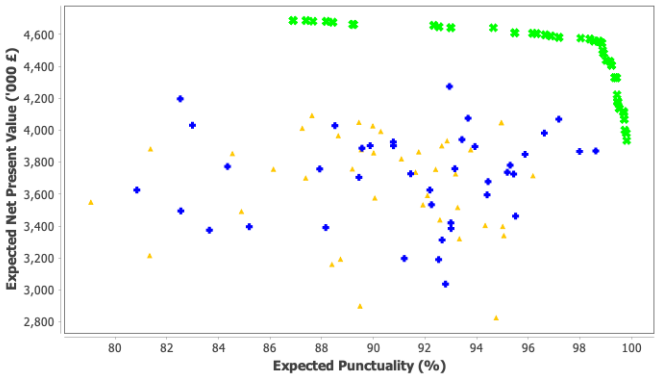
(b) Release Planner



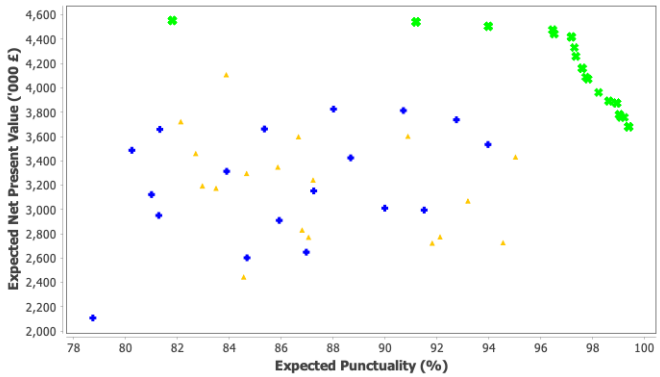
(c) Word Processor



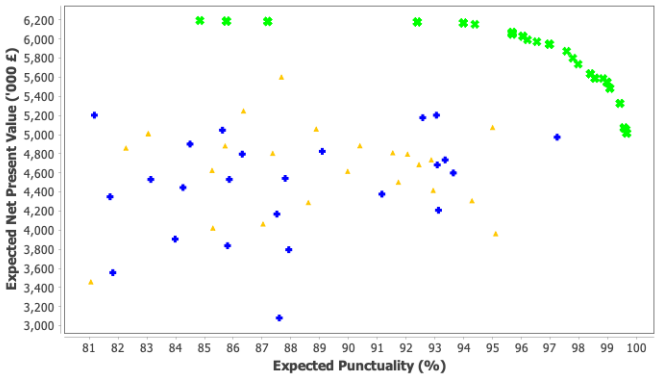
(d) RALIC



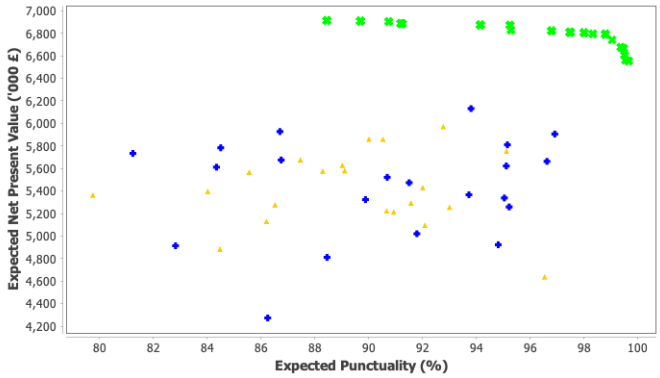
(e) Synthetic-30



(f) Synthetic-50

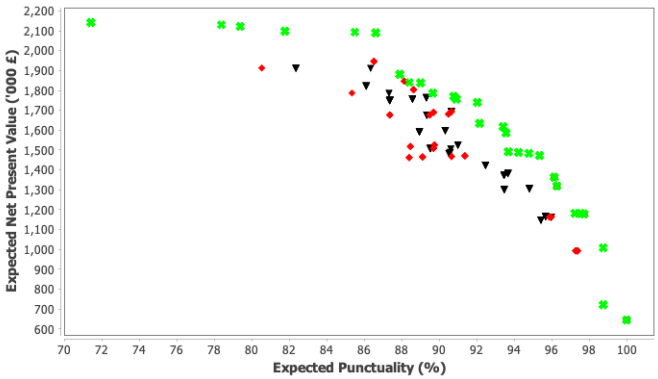


(g) Synthetic-30

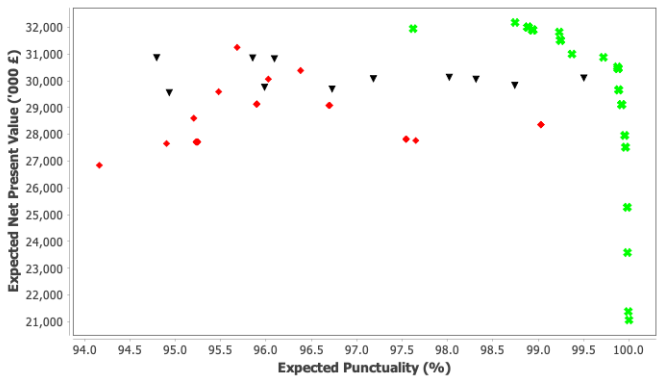


(h) Synthetic-50

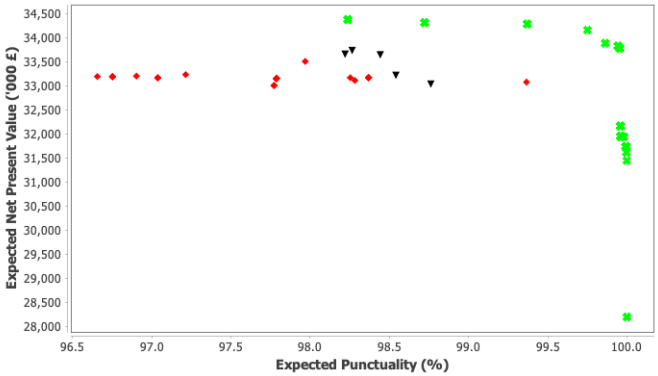
Fig. 5: Examples of shortlists generated by BEARS (green crosses), EVOLVE-II (blue diamonds) and BEARS-deterministic (yellow triangles). These examples correspond to the first runs out of 30.



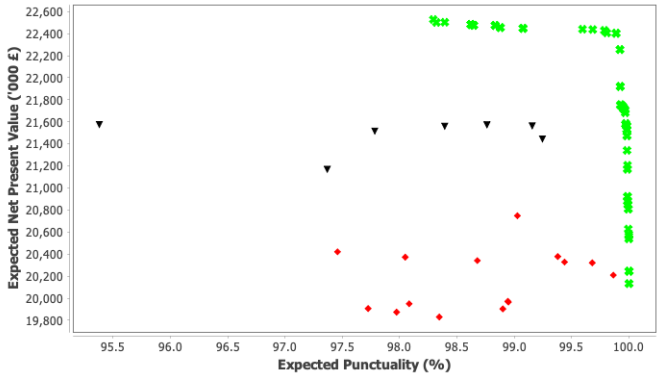
(a) Local Government



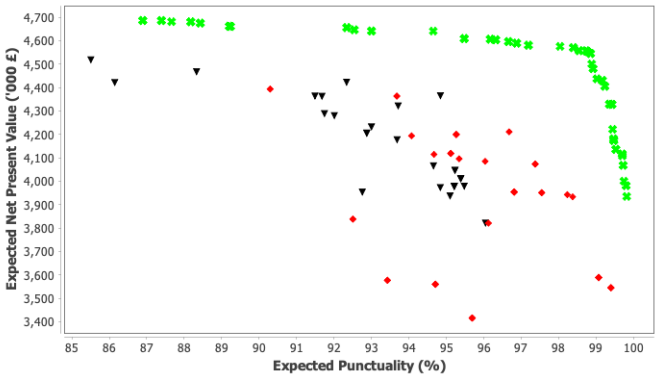
(b) Release Planner



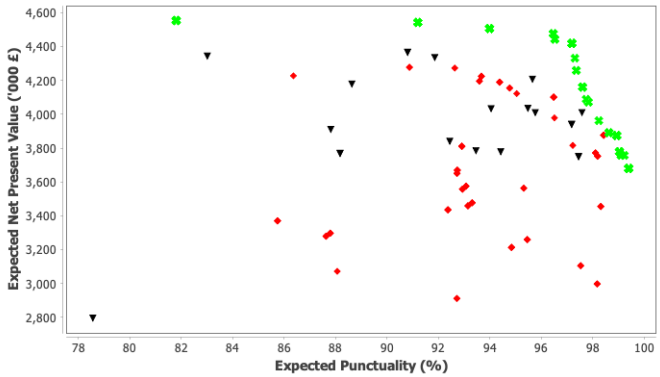
(c) Word Processor



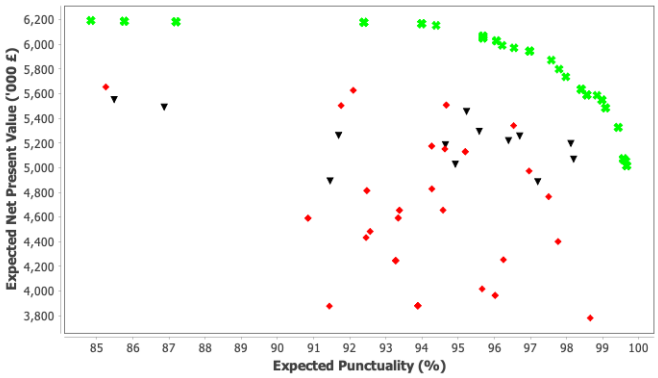
(d) RALIC



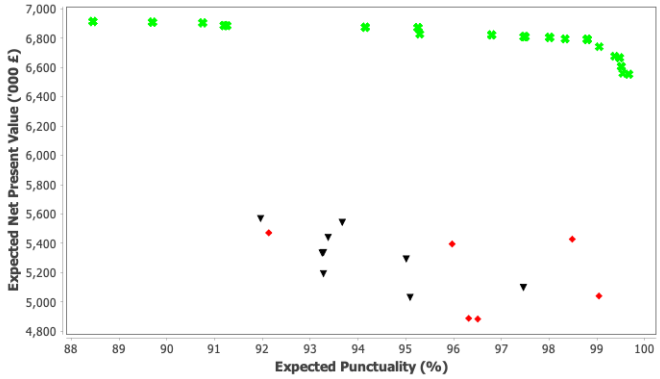
(e) Synthetic-30



(f) Synthetic-50



(g) Synthetic-30



(h) Synthetic-50

Fig. 6: Examples of shortlists generated by BEARS (green crosses), EVOLVE-with uncertainty (red diamonds) and BEARS-fixed-scope (black triangles). These examples correspond to the first runs out of 30.

TABLE 9: Release planning methods' run-times in seconds

Product Backlog	H	EVOLVE-II	Bears-deterministic	Evolve-with-uncertainty	Bears-fixed-scope	BEARS
Local Government Project	2	2	3	10	18	45
	3	2	4	12	20	48
	4	3	4	15	21	60
	5	3	5	17	25	65
Release Planner	2	2	3	10	15	37
	3	2	4	14	25	55
	4	3	5	16	30	70
	5	5	8	20	35	95
Word Processor	2	2	4	18	36	95
	3	3	7	20	47	135
	4	4	8	23	58	155
	5	6	10	26	65	163
RALIC	2	4	10	37	85	153
	3	5	12	45	102	192
	4	5	12	54	123	225
	5	7	15	68	150	259
Synthetic-30	2	2	6	15	33	75
	3	3	7	17	42	95
	4	3	9	17	40	112
	5	5	11	20	49	125
Synthetic-50	2	5	9	20	68	158
	3	7	11	25	75	170
	4	7	12	29	84	188
	5	9	14	37	96	205
Synthetic-100	2	6	13	56	75	229
	3	6	16	75	94	284
	4	7	20	85	110	302
	5	7	24	97	115	357
Synthetic-200	2	9	20	83	112	288
	3	9	21	85	122	365
	4	10	22	95	130	486
	5	10	30	120	150	533

TABLE 10: Performance of a random search and the 3 Multi-Objective Optimisation Algorithms used in BEARS.

		Mean Hypervolume				Mean IGD+			
Product Backlog	H	NSGA-II	SPEA2	MOCell	Random	NSGA-II	SPEA2	MOCell	Random
Local Government Project	2	0.956	0.947	0.903	0.825	0.019	0.011	0.009	0.080
	3	0.982	0.982	0.991	0.878	0.076	0.042	0.002	0.115
	4	0.973	0.988	0.968	0.943	0.021	0.004	0.002	0.110
	5	0.975	0.982	0.996	0.888	0.019	0.009	0.005	0.114
Release Planner	2	0.806	0.807	0.851	0.752	0.268	0.025	0.019	0.444
	3	0.873	0.949	0.879	0.820	0.424	0.088	0.020	0.635
	4	0.948	0.973	0.900	0.917	0.178	0.014	0.005	0.301
	5	0.945	0.927	0.969	0.739	0.047	0.026	0.001	0.180
Word Processor	2	0.938	0.946	0.949	0.820	0.560	0.138	0.032	1.880
	3	0.971	0.988	0.999	0.907	0.257	0.016	0.005	1.050
	4	0.971	0.986	0.994	0.915	0.076	0.035	0.001	0.525
	5	0.925	0.938	0.996	0.925	0.052	0.006	0.005	0.361
RALIC	2	0.984	0.986	0.996	0.780	0.038	0.001	0.002	6.040
	3	0.967	0.992	0.994	0.686	0.319	0.002	0.081	9.700
	4	0.927	0.971	0.997	0.669	0.455	0.005	0.008	6.260
	5	0.922	0.984	0.942	0.660	0.466	0.019	0.001	4.320
Synthetic-30	2	0.594	0.598	0.517	0.526	0.579	0.509	0.025	0.726
	3	0.720	0.723	0.721	0.670	0.383	0.251	0.378	0.412
	4	0.846	0.947	0.845	0.818	0.284	0.008	0.039	0.353
	5	0.933	0.945	0.955	0.906	0.021	0.004	0.01	0.125
Synthetic-50	2	0.718	0.955	0.714	0.688	0.216	0.011	0.161	0.216
	3	0.943	0.945	0.970	0.808	0.123	0.011	0.014	0.137
	4	0.958	0.962	0.889	0.914	0.030	0.009	0.007	0.161
	5	0.963	0.968	0.965	0.841	0.034	0.008	0.012	0.332
Synthetic-100	2	0.886	0.886	0.687	0.711	0.136	0.058	0.134	0.137
	3	0.936	0.936	0.888	0.931	0.037	0.016	0.012	0.104
	4	0.948	0.948	0.948	0.758	0.083	0.014	0.039	0.265
	5	0.922	0.922	0.959	0.839	0.083	0.025	0.024	0.506
Synthetic-200	2	0.726	0.709	0.727	0.681	0.023	0.021	0.021	0.024
	3	0.975	0.945	0.961	0.956	0.024	0.008	0.020	0.081
	4	0.970	0.911	0.953	0.887	0.061	0.003	0.048	0.274
	5	0.970	0.964	0.938	0.819	0.079	0.009	0.047	0.606
Overall		0.726	0.829	0.784	0.670	0.020	0.003	0.002	0.110

TABLE 11: Statistical Significance (p-value) of the differences between MOEAs on BEARS release planning problems using Mann-Whitney U test. Highlighted cells are those where the differences are not satisfically significant (p>.05)

Product Backlog	Metric	H	Random vs. NSGA-II	Random vs. SPEA2	Random vs. MOCell	NSGA-II vs. SPEA2	NSGA-II vs. MOCell	SPEA2 vs. MOCell
Local Government Project	HV	2	$2.87E-11$	$2.87E-11$	$8.68E-01$	$2.44E-03$	$2.87E-11$	$2.87E-11$
		3	$6.87E-08$	$5.45E-09$	$3.85E-08$	$4.43E-04$	$1.89E-08$	$9.76E-06$
		4	$3.88E-04$	$8.01E-06$	$5.87E-09$	$8.08E-01$	$6.11E-06$	$8.86E-04$
		5	$2.84E-05$	$5.43E-10$	$2.87E-11$	$2.83E-03$	$3.50E-08$	$5.62E-07$
	IGD+	2	$2.87E-11$	$3.96E-09$	$2.87E-11$	$9.62E-03$	$7.14E-08$	$5.70E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.98E-06$	$3.88E-11$	$1.43E-10$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.09E-05$	$9.44E-11$	$2.08E-06$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.47E-07$	$2.26E-10$	$1.11E-07$
Release Planner	HV	2	$3.96E-07$	$2.87E-11$	$5.32E-10$	$9.06E-01$	$4.79E-05$	$7.44E-09$
		3	$5.27E-06$	$9.44E-11$	$1.38E-05$	$3.50E-08$	$2.25E-01$	$1.47E-01$
		4	$7.10E-04$	$8.12E-09$	$2.14E-01$	$7.43E-05$	$2.19E-02$	$3.66E-07$
		5	$4.22E-05$	$4.27E-06$	$3.88E-11$	$7.01E-01$	$4.44E-02$	$1.09E-03$
	IGD+	2	$3.31E-10$	$6.81E-09$	$2.87E-11$	$1.37E-08$	$3.51E-11$	$1.65E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.63E-08$	$3.06E-09$	$1.63E-04$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.49E-08$	$2.74E-10$	$8.71E-01$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.99E-07$	$5.32E-10$	$5.22E-09$
Word Processor	HV	2	$1.02E-09$	$2.87E-11$	$2.87E-11$	$8.14E-03$	$1.47E-02$	$1.53E-02$
		3	$3.64E-10$	$2.87E-11$	$2.74E-10$	$7.10E-04$	$5.32E-10$	$2.98E-06$
		4	$6.07E-06$	$1.77E-09$	$9.31E-10$	$8.63E-02$	$3.21E-08$	$2.40E-06$
		5	$2.87E-01$	$4.34E-04$	$1.37E-04$	$1.17E-01$	$5.79E-05$	$2.87E-02$
	IGD+	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.13E-09$	$2.87E-11$	$1.12E-09$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$6.81E-09$	$2.87E-11$	$1.54E-10$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.71E-08$	$5.77E-11$	$7.04E-10$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.31E-10$	$3.51E-11$	$5.10E-02$
RALIC	HV	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.13E-02$	$4.27E-06$	$2.82E-03$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.80E-07$	$1.94E-09$	$4.69E-01$
		4	$7.73E-10$	$2.87E-11$	$2.87E-11$	$1.15E-06$	$2.49E-10$	$3.70E-06$
		5	$5.32E-10$	$2.87E-11$	$2.87E-11$	$1.02E-09$	$1.94E-02$	$5.32E-10$
	IGD+	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.35E-09$	$6.37E-11$	$9.27E-03$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.49E-10$	$1.54E-10$	$7.44E-09$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.62E-09$	$6.37E-11$	$5.65E-02$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.62E-09$	$7.03E-11$	$9.44E-08$
Synthetic-30	HV	2	$1.17E-01$	$3.88E-04$	$7.78E-03$	$5.35E-01$	$2.76E-02$	$7.36E-02$
		3	$5.32E-10$	$1.12E-09$	$2.87E-11$	$2.14E-01$	$1.53E-02$	$8.48E-01$
		4	$6.51E-06$	$2.87E-11$	$2.87E-11$	$1.84E-04$	$1.47E-01$	$1.14E-01$
		5	$6.04E-02$	$2.82E-03$	$2.19E-04$	$3.83E-01$	$9.19E-02$	$3.59E-01$
	IGD+	2	$4.28E-07$	$2.26E-10$	$2.87E-11$	$3.21E-02$	$3.31E-10$	$1.95E-07$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.95E-07$	$5.43E-05$	$3.26E-05$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.73E-10$	$1.77E-08$	$1.05E-05$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$5.84E-10$	$1.49E-08$	$6.98E-05$
Synthetic-50	HV	2	$5.77E-11$	$5.23E-11$	$7.49E-04$	$8.70E-08$	$1.10E-02$	$1.77E-08$
		3	$7.44E-09$	$1.37E-08$	$9.31E-10$	$2.49E-01$	$1.47E-02$	$5.10E-02$
		4	$6.16E-05$	$1.48E-09$	$2.76E-04$	$7.34E-01$	$5.70E-03$	$3.45E-06$
		5	$1.62E-09$	$2.87E-11$	$2.87E-11$	$1.60E-01$	$3.09E-02$	$6.36E-01$
	IGD+	2	$9.64E-01$	$4.40E-10$	$3.18E-11$	$3.66E-09$	$2.26E-10$	$9.44E-08$
		3	$1.11E-07$	$2.87E-11$	$2.49E-10$	$3.31E-10$	$2.68E-07$	$9.41E-01$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.02E-09$	$3.88E-11$	$1.73E-02$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.74E-10$	$1.04E-10$	$6.04E-02$
Synthetic-100	HV	2	$3.71E-05$	$3.71E-05$	$1.20E-07$	$9.99E-01$	$6.56E-05$	$6.56E-05$
		3	$6.05E-01$	$6.04E-01$	$1.02E-07$	$9.99E-01$	$3.33E-02$	$3.33E-02$
		4	$3.51E-11$	$3.51E-11$	$3.88E-11$	$9.99E-01$	$9.76E-01$	$9.76E-01$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$6.73E-04$	$6.73E-04$
	IGD+	2	$1.31E-07$	$1.31E-07$	$2.26E-10$	$9.99E-01$	$3.09E-04$	$3.09E-04$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$1.60E-01$	$1.60E-01$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$4.58E-04$	$4.58E-04$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$9.41E-01$	$9.41E-01$
Synthetic-200	HV	2	$8.12E-09$	$7.79E-03$	$2.87E-11$	$3.91E-01$	$2.37E-02$	$2.14E-01$
		3	$5.71E-09$	$1.73E-02$	$2.87E-11$	$8.01E-06$	$2.68E-05$	$2.07E-04$
		4	$4.49E-08$	$4.79E-05$	$2.87E-11$	$6.28E-07$	$1.20E-07$	$4.58E-04$
		5	$1.54E-10$	$2.87E-11$	$2.87E-11$	$7.34E-01$	$2.56E-02$	$4.10E-04$
	IGD+	2	$1.53E-02$	$2.87E-11$	$4.73E-09$	$1.54E-10$	$3.50E-08$	$2.04E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.74E-10$	$1.02E-07$	$3.96E-05$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.76E-11$	$1.63E-08$	$2.10E-08$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.17E-11$	$1.23E-09$	$5.22E-09$
Overall	HV		$6.17E-98$	$2.84E-139$	$1.09E-110$	$2.71E-11$	$3.29E-06$	$3.40E-01$
	IGD+		$9.03E-110$	$1.74E-243$	$5.99E-246$	$2.30E-98$	$6.82E-105$	$9.85E-01$